AUTOMATIC DETECTION OF ABNORMAL MOTION

A Thesis
Presented to
The Academic Faculty

By

Daniel J.G. Dichek

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2018

AUTOMATIC DETECTION OF ABNORMAL MOTION

Approved by:

Dr. Fumin Zhang, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Sam Coogan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Britney Schmidt
School of Earth and Atmospheric Science
Georgia Institute of Technology

Date Approved: April 11, 2018

ACKNOWLEDGEMENTS

I am grateful for the support my parents provided through out my time as a graduate student. I would also like to thank Dr. Vela for allowing me to borrow one of his Turtlebot 2 development platforms for an extended period of time, and Chad Ramey for answering software and networking questions.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

Modern robots are almost helpless in the real world despite their unprecedented perception. They struggle to react appropriately to random events. They require human supervisors to detect and react to situations for which they are not explicitly prepared. To survive in the real world without human supervision, robots must autonomously distinguish normal and abnormal situations. The abstract nature of this task has no simple qualitative solution, which prohibits an algorithmic solution and mandates human intervention. This thesis presents a method for robots to autonomously identify abnormal motions. The method requires a human operator to guide a robot through the motions it is expected to perform during normal operation, which allows the robot to group recurring motions with a clustering algorithm. The resulting clusters form a quantitative model of "normal motion". This model allows the robot to calculate the probability that an observed motion is a normal motion. Results demonstrated that abnormal motions such as being lifted, pushed or stepped on, were correctly indicated with low probabilities, and that motions experienced during normal operation were classified as highly probable. This thesis also describes an independent system for robot motion observation. This independent system allows rapid implementation of motion-analysis algorithms on any robot.

CHAPTER 1

INTRODUCTION

It is extremely difficult for modern robots to retain control of themselves in situations for which they are not explicitly prepared. New approaches to robot control that incorporate machine learning algorithms retain control by adapting to uncertainties that would typically cause instabilities [1, 2, 3]. However, these controllers have solved only motion control problems such as achieving sequences of vehicle position and velocity goals. They do not solve the more complex problem of deciding whether those position and velocity goals *should* be achieved. Robots must be able to use what they perceive to decide whether a command should be executed without human assistance. This sentiment is echoed in a recent paper on the necessity and implementation of such abilities in robots [4].

Current methods for abstract decision-making in mobile robots rely on extensive preparation and training of the robots by experts. Even with expert training, robots still fail to operate safely even in slightly unusual situations [5]. Examples of highly advanced but dramatically inept robots abound [5, 6]. In most of these examples, the robots became hazardous because they lost control. One could argue that these robots not have become

Figure 1.1: Robot defeated by a slight abnormality (wet tiles)

hazardous if they had more stable controllers.

The idea that improved controllers will make robots more safe can be dismissed with the case of the 300-pound "public safety robot" that struck and ran over a toddler in a mall (who survived with minor bruising), despite the attempts of his parents to interfere [7]. Here is an example of a robot with controllers that met their position and velocity goals and maintained stability despite the disturbances introduced by a struggling toddler and two adults. It is an excellent model of a robot that was dangerous not due to instability, but because it failed to perceive that something was wrong with its actions. By achieving its position and velocity goals, the robot was also unhelpful in achieving its primary goal of improving public safety. Examples abound of robots becoming dangerous as a result of their attempts to maintain control [8, 9, 10, 11].

All of the robot failures referenced so far have featured robots with carefully designed and tested collision-avoidance systems. Despite the effort devoted to perfect these systems, they failed embarrassingly. Robust systems for detecting and preventing hazardous operation of motor vehicles do exist. 1) the Antilock Braking System (ABS) in most modern vehicles; and 2) the stall warning system which are required by federal law on all modern commercial aircraft (CFR § 23.207) [12, 13]. Although these systems are robust and demonstrably nearly infallible, their excellence is the product of many decades of research, development and testing. However, these systems only solve very specific problems. Both systems also rely on humans to perceive and react to abnormalities. ABS requires humans to apply brakes. Stall warning systems require pilots to intervene to prevent stalling. And even with decades of safety system development, the most advanced warning and safety systems in airplanes are still not trusted over pilots. In a video taken during the Airbus A380 flight test program, pilots are shown growing uncomfortable with an unannunciated abnormality (aeroelastic fluttering), and avoid catastrophic loss of the airplane by landing promptly [14].

It is essential that robot control systems include robust safety systems because robots

are invading human environments earlier and in greater quantities than cars and airplanes did in their early stages of development. Mass-production techniques were originally developed to accelerate the production of automobiles [15] and airplanes. Now that mass manufacturing technologies are mature, other machines, such as robots, can be rapidly mass-produced and introduced into daily human life before their safety can be guaranteed. There are already examples of fatal incidents involving unproven collision-avoidance technologies [9]. Currently, there is no substitute for the human ability to detect anomalies, so modern robot safety systems rely on human supervisors. Even in California, where the requirement for a driver to be in an autonomous vehicle was recently removed, (§ 227.02 (b) of the California Code of Regulations Title 13, Division 1, Chapter 1, Article 3.7 - Testing of Autonomous Vehicles), still "requires a human test driver or a remote operator to continuously supervise the vehicles performance of the dynamic driving task" [16].

This research investigated a way to automatically detect motion abnormalities that would typically require a human supervisor: instead of training a robot to detect specific problems, allow it to build a model of "normal behavior" that can be used to calculate the probability that an observed motion is "normal". This probability can be used independently or combined with other probabilistic systems in the vehicle to justify initiation of a safety maneuver.

Specifically, the goal of this research is to develop a framework for automatically detecting abnormal motions on mobile robots. Ideally, the abnormal motion detection framework can be installed on any robot with satisfactory hardware. To calculate the probability that an observed motion is normal, the robot must generate a model that will allow quantitative comparison of motion measurements. This thesis assumes that robot construction and control generate repeatable patterns of motion, termed "symptomatic motions" and explained in section 2.4, that can be used to generate a model of normal operation. Given a model of normal motion, an abnormal motion detection algorithm should be able to distinguish between normal and abnormal motions without human supervision.

Two systems were developed to achieve the research goal. The first system automated observation of mobile robot motion. The benefits of an independent robot observation platform over interfacing with existing robot sensors are explained in section 2.5. The robot observation platform was used to quickly generate datasets and enabled cross-platform algorithm development and data analysis. The second system implemented an algorithm for automatic abnormal motion detection. This system uses datasets generated by the robot observation system to guide algorithm development, and to generate models of normal motion.

The algorithm was tested with motion datasets generated by an iRobot Roomba 690 robotic vacuum cleaner and a Turtlebot3 Burger mobile robot development platform. Datasets varied between entirely normal operation, in which the robot performed a repetitive job in a controlled environment without disruption, and abnormal operation, in which the robot was intentionally disrupted during its job.

The data collected by the robot motion observation system shows that patterns in accelerations can be autonomously recognized and used to build an expectation of "normal motion". The abnormal motion detection algorithm used a clustering algorithm to identify and model these patterns, and to perform comparisons between known acceleration patterns and observed accelerations. When tested with previously unseen data, the algorithm correctly indicated instances of (abnormal) motion introduced by the researcher by marking them as improbable, and did not mistakenly classify periods of normal operation as abnormal. The results are preliminary and cannot yet be compared qualitatively with other methods.

CHAPTER 2

BACKGROUND

Autonomous vehicles are rapidly invading human environments despite their poor ability to detect and react appropriately to abnormal situations. Hobbyist drone user registrations with the FAA have grown quickly, and currently exceed 800,000 [17]. Drone incidents have increased in turn: The FAA reported a 46% increase in the number of drone incidents at airports between 2015 and 2016 [18]. The Google Autonomous Car Project, now operating as Waymo, plans to add 20,000 autonomous vehicles to their fleet of test vehicles [19]. In compliance with California State Law, Waymo reported every instance of human disengagement of vehicle autopilot during 2017: 63% of these disengagements were due to an unwanted behavior, perception discrepancy, or incorrect prediction of extravehicular behavior rather than software or hardware failures [20]. Figures reported by Cruise, another autonomous vehicle company, also show that a significant number of disengagements were not due to equipment failure, but rather to issues with vehicle planning or perception [21]. Currently, humans are required by law to detect and respond to anomalies in these robots [16]. A system that could automatically detect abnormal motions in robots could wean robots from human supervision.

2.1 Modeling with Machine Learning Algorithms

Machine learning algorithms allow automatic generation of classification and regression algorithms by generalizing rules inferred from exemplar data [22]. Models built with machine learning algorithms allow robots to learn to recognize previously-demonstrated motions, and then to use those to characterize new motions [23, 24]. Researchers have demonstrated a helicopter that learned its own complicated dynamics and successfully executed complex aggressive maneuvers [1, 2]. Both papers note that the difficulty in controlling a

helicopter resides in the presence of unobserved state variables. The machine learning algorithms used in both papers achieve impressive performance by inferring the unobserved variables. To obtain the models needed to perform advanced maneuvers, the researchers first had to execute and record these maneuvers.

Instead of executing and recording motions, it is possible to obtain vehicle dynamical models by simulation, also known as System Identification. However, simulations suffer from being fundamentally disconnected from reality. They can allow actions that are impossible in the real world such as instantaneous change in continuous values, or produce believable results for the wrong reasons. The formal name for the discrepancy between simulation and reality is known as the Reality Gap [25]. A recent publication from Georgia Tech introduces a machine-learning assisted solution to the reality gap: a two-component algorithm called UP-OSI [26]. The researchers describe a Universal Policy (UP) controller that is generic enough to control a given set of dynamic models. These dynamic models are supplied by an Online System Identification (OSI) algorithm that predicts the dynamic model parameters with recent motion measurements. The UP-OSI algorithm was able to estimate the time-varying dynamic models of the systems it controlled without prior training by simulating its motion under different conditions and pre-computing optimal control policies for each condition. However, the researchers note that it is limited to low-dimensional systems ($\mathbb{R}^4$), and many real world dynamical systems require high-dimensional representations to be controlled effectively.

Models capable of abstract decision-making can be built and autonomously improved with machine learning. For example, researchers demonstrated the ability of a robot hand to incrementally learn and differentiate between grasped objects [27]. This task is abstract because it requires a distinction to be drawn between objects without easily quantifiable differences: what is the difference between grabbing an empty can and a half-full bottle of lotion? The answer cannot be effectively captured with a single number. The researchers used a sparse online infinite echo-state Gaussian process [28] to incrementally and automat-

ically build an abstract, multivariate, time-varying model of its tactile sensations for each object. This allowed the objects to be differentiated quantitatively. Here, the dynamic models being inferred were the tactile properties of objects being grasped, which had relatively high-dimensional features ($\mathbb{R}^{25}$).

Machine learning algorithms show a strong ability to generate models for system and object identification. Robots might rely less on human supervision in uncertain circumstances with a system that can model "normal" behavior and use that model to identify "abnormal" behaviors. Machine learning algorithms may be able to generate abstract models that could be used to distinguish between normal and abnormal motions.

2.2 Action Recognition with Machine Learning

The idea of recognizing separate "actions" or "motions" has been very popular, but only if the actions being recognized are human. Of all the references cited in a "Review on action recognition and mapping", published in a leading robotics journal, none of them referred to recognizing the actions of a robot [29]. As an informal survey, even the Wikipedia page on Machine Learning Datasets lacks a set of robot actions - only datasets with human actions are available [30]. One could argue that robots already observe their actions because they command them, and this is indeed the basis for many anomaly detection methods [31, 32]. Ideally, an anomaly detection system would not need to depend on information internal to a robot controller in order to detect anomalies. For example, a variety of papers have demonstrated the ability to differentiate between human motions and gestures with accelerometers [33, 34, 35, 36]. None of these systems relied on human neural signals (the human equivalent of motion commands) or models of human motion. Each of these papers present "offline" methods that require repeated prior demonstration of an action in order to classify it properly.

Avigilon, a security system company, demonstrated an "abnormal motion detection" system that detects abnormal human behavior in video [37]. It does not require users to

provide motion models or internal information from the subject it analyzes for abnormal motion. Similarly, a research group demonstrated a system for real-time detection of abnormal motion in video. The system developed a model of "normal" by training an algorithm with video of normal pedestrian traffic and then tested performance with videos of abnormal events, such as cars driving on the sidewalk or people running quickly [38]. Another popular and successful technique for recognizing actions with machine learning is motion segmentation, the process of identifying when actions begin and end. Motion segmentation has been used to segregate motions performed by physical therapy patients for inter-patient comparison [39]. Motion segmentation has also been used to help robots learn to perform tasks [40]. Robots would benefit from a similar ability to segment their own motions and compare them to "exemplar" motions.

2.3 Anomaly Detection

Anomaly detection is the process of identifying unusual trends in data and is used in many different fields to great benefit [41, 42]. (V. Chandola et al. 2009) provides an excellent survey of anomaly detection techniques and their applications [43]. In industrial control applications, abnormal motion detection is known as "condition monitoring". Specifically, condition monitoring is the process of monitoring a parameter such as vibration to detect the onset of a potential mechanical issue in a machine [44]. Fluke, a major test equipment manufacturer, sells a system for instrumenting factory machinery and monitoring for developing faults [45]. Another major manufacturing company, Valmet, sells condition-monitoring sensors and software [46]. Many papers have been published demonstrating the utility of vibration measurements in predicting mechanical faults [47, 48, 49, 50]. A report from Honeywell describes the need for anomaly detection in its opening paragraph:

> *Abnormal Situations comprise a range of process disruptions in which petrochemical plant personnel must intervene to correct problems with which the control systems can not cope. Preventable losses from abnormal situations cost the U.S. economy at least $20B annually.* [51]

The report describes a process implemented with teams of people to monitor conditions and report anomalies. Condition monitoring and anomaly detection should be an autonomous function of an industrial process to reduce the burden on human operators. Simply applying condition monitoring sensors to robots would allow detection of anomalies, albeit very specific ones. Ideally, the machine learning techniques used in robotics could be used to generalize condition monitoring techniques. Progress towards this goal is discussed in the following section.

2.3.1 Anomaly Detection in Robots

Recent robotics research has addressed abnormal motion detection in robots. Sometimes this is referred to as "execution monitoring", which implies knowledge of the motion being executed. (Pettersson 2005) provides a thorough although dated survey of execution monitoring in robotics that also illustrates its rarity in robotics research relative to other fields [52]. Most methods for anomaly detection in robots assume that the robot maintains an awareness of the action it executes, and an expectation of what it should experience while performing that action.

A recent paper proposing an algorithm for anomaly detection in underwater vehicles leverages machine learning algorithms to model robot motion in uncertain circumstances, and this is used to generate an expectation of "normal" motion [31]. This method relies on expert-provided maximum and minimum velocity thresholds, a motion model of the vehicle and of the forces applied to the vehicle by fluid flow, and only examines measured and estimated velocity to provide anomaly detection. Ideally, an anomaly detection algorithm

would learn a model of "normal motion" with minimal human guidance. Additionally, a vehicle with many sensors should not risk ignoring other sensors in favor of one measurement. As an alternative to using a single measurement as an indicator of normality, [32] demonstrates the advantages of using multi-modal (haptic, auditory, visual and kinematic) sensor signals for execution monitoring.

A similar paper suggests a way to reduce dependence on robot-specific parameters (robot motion models which require access to robot-internal control data, motion models which require estimation of the effects of external forces on the vehicle) by presenting object-centric probabilistic models of applied forces [53]. The independence from robot-specific information presented by the method in [53] allowed different robots and even humans to help build models useful for anomaly detection in robots. Furthermore, the probabilistic output of the anomaly detection framework in [53] allows easy integration with other probabilistic anomaly detection algorithms: computing simple joint or conditional probabilities would be simple, and rigorous mathematical frameworks for combining many probabilistic signals are well-studied and available.

An ideal system for abnormal motion detection in robots would allow multi-modal sensor inputs while maintaining a self-contained (platform-independent) implementation, the advantages of which are demonstrated in [32] and [53]. Additionally, the anomaly detector would be able to autonomously discover patterns in the data. This can be accomplished with clustering, and a recent clustering algorithm has succeeded where others have not by being able to hierarchically cluster data [54]. The ideal system would also feature a probabilistic output to facilitate combination with other anomaly detection systems.

2.4 Symptomatic Motions

Automatic abnormal motion detection relies on a key assumption: Robots that perform repetitive jobs in a controlled environment will experience a finite set of motions that are symptoms of how they are built and how they perform their job. These symptomatic mo-

tions can be measured with enough fidelity to generate probabilistic models which can be used to calculate the probability that an observed motion is "normal" or not.

2.4.1 Abnormal Motion Detection by Clustering Symptomatic Motions

A robot that repeats a clearly defined job, such as cleaning a room, has a set of fundamental motions it repeats to complete its job. Every time the robot repeats these motions, it undergoes the same forces in the same sequences, with some finite variance. For example, to move through a room, a cleaning robot must frequently drive forward and turn. To do this, the robot generates a sequence of target velocities to follow and then changes between them with a linear change in velocity. That linear change in velocity is consistent and produces one example of a symptomatic motion: a constant acceleration, generated every time the robot changes its velocity. Even if the change in velocity is noisy or exponential, the robot will still experience a relatively constant or linear acceleration.

Other examples of symptomatic motions of robots with repetitive jobs include sequences of accelerations, spectral density of motion frequency components, and constrained magnitudes or directions of measured forces. Each of these symptomatic motions are produced by robot- or job-specific behaviors and should be consistent enough between jobs to use as a measure of the state of the robot or job. The set of accelerations experienced while vacuuming a room, while somewhat random, but within bounds, is obviously different from the set of accelerations experienced while being thrown across the room.

Clustering is the process of grouping similar data points. "Similarity" in data is typically easy for humans to see, but difficult to describe mathematically. Because clustering algorithms are supposed to detect structure in data for humans, they are typically considered a form of unsupervised learning. There are "supervised" clustering algorithms, such as k-means, which require the specification of the number of clusters to be generated.

Ideally, a clustering algorithm would autonomously separate data into useful clusters. Many clustering algorithms perform a variation of the following process: group data points

by calculating sample density with respect to area, choosing centroids of points in the dataset and considering them "centers" of clusters, and then labeling all data within a certain radius of the center of a cluster as being part of that cluster. For example, meanshift clustering "shifts" data points towards each other based on density gradients in order to form separate centroids. All points that get "shifted" into a particular centroid are labeled as being a part of that centroid's cluster. However, density-based methods do not allow for the separation of complex structures such as concentric or overlapping clusters.

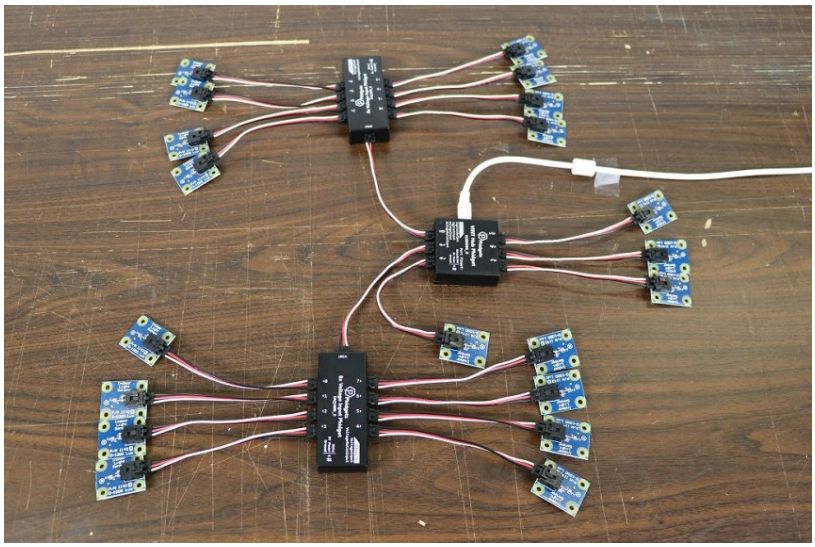
A recent paper proposes a solution to the shortcomings of density-based clustering by allowing a representation of cluster hierarchy [54]. Density-based clustering works well, but sometimes merges clusters that should be separate. If these clusters can be labeled as "similar but different" at some point before they are merged during the algorithm execution, the hierarchy of clusters can be extracted, allowing separation of complex cluster structures. HDBSCAN implements hierarchical density-based clustering.

Consider how modeling these symptomatic motions could simplify the detection of complex issues. It might be difficult for a robot to detect being lifted up and moved without data from a demonstration, but the robot should have never experienced anything like that on its own. It is difficult to detect wheels slipping. Special algorithms must be used to detect and compensate for slipping. The forces experienced during a wheel slip should clearly be unusual when compared to a comprehensive model of "normal" forces experienced by wheels. This is how Automatic Abnormal Motion Detection can detect abnormal motions without prior demonstration: by building a model of normal motions in such a way that it can compare observed motions against its model and generate a quantitative comparison.

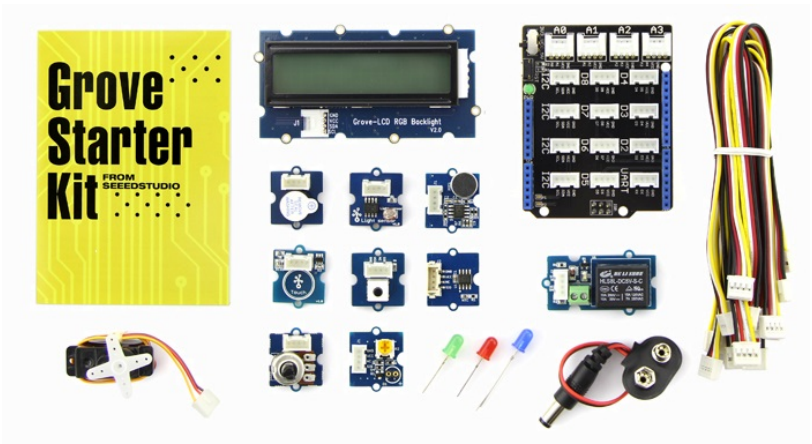
2.5 Robot Motion Observation

Robots would benefit from a self-contained platform that allowed repeatable and comparable motion measurements. Ideally, the system would be unobtrusive and easy to install on any robot. Such a system would allow quick implementation of multi-modal motion-

analysis algorithms with outputs that could be compared across robots, such as the ones mentioned in [53, 32]. Early forms of such a system already exist in the form of kits that allow the formation of arbitrary sensor networks [55, 56]. Both of these kits are hardware platforms, and although they provide software interfaces, they do not provide a software platform for performing motion analysis given sensor data.



(a) Phidget^{TM} sensor network



(b) Seeed Studio Grove^{TM} sensor network kit

Figure 2.1: Commercial off-the-shelf kits for independent arbitrary sensor networks.

Similar self-contained motion-data aggregation systems are mandatory on airplanes and helicopters in the form of FDRs (Flight Data Recorders) and HUMS (Health and Usage Monitoring System) [57, 58]. Major aerospace companies have improved airplane perfor-

mance and reliability over many decades by performing a slow, manual implementation of abnormal motion detection when they review the data from an aircraft incident and decide how to react to it.

An independent platform for motion observation would also facilitate experimentation with sensors that are not commonly used in robotics, but that have shown promise in other areas. There are many "unusual" sensors a researcher may wish to investigate for their potential in future robotic systems. Two examples include an optical sensor for flight control based on an organ present in house flies [59], and a system of optical sensors for navigation inspired by organs found in cockroaches [60]. Biologically inspired sensors may provide many conspicuous advantages over conventional sensors [61], although it is currently difficult to justify their incorporation in current designs.

There is clear merit in using machine learning algorithms to model robot motion: these models allow more robust control of vehicles, and recent work has shown that these models can also be used to detect anomalies. Autonomously detecting motion anomalies will help vehicles become independent of human supervision, but not if they require copious human expertise to implement. Ideally, an anomaly detection system on a robot would allow autonomous probabilistic modeling of "normal motion" as perceived by multimodal sensors that could be combined with models generated by other robots. This work explores the application of a relatively new hierarchical clustering algorithm, HDBSCAN, to probabilistic anomaly detection, implemented on a novel robot-independent motion observation platform.

CHAPTER 3
TECHNICAL APPROACH

Two systems were developed to implement automatic abnormal motion detection. The first system independently observes robot motion. The novel aspects of this system are its affordances for scaling, rapid deployment, and integration of multimodal sensors. The system also differs from many others by having no integration requirements: it is self-contained and only needs to be mechanically adhered to a robot under observation. This accelerates data collection by minimizing setup time and cost. This motion observation system was used to rapidly collect and analyze motion data with minimal overhead.

The second system implements automatic abnormal motion detection in mobile robots. It was developed with the robot motion observation system. The datasets from the motion observer were used to evaluate the performance of the algorithm.

Both the observation system and the abnormal motion detection algorithm use commonly found and "research friendly" (open-source, well-documented and cheap or free) hardware and software such as Ubuntu Linux, the Robot Operating System (ROS), and Raspberry Pi embedded computers. No proprietary software was used to implement or evaluate the algorithm.

3.1 Independent System for Robot Motion Observation

The purpose of the independent system for robot observation was to accelerate collection of robot data by minimizing delays caused by setting up and integrating the observation systems into an existing robotic platform. This was accomplished by aggregating common and reliable off-the-shelf (non-custom) tools with thorough documentation and large communities of helpful users. The diagrams in figures 3.4 and 3.7 illustrate the components of the observation system.

Many robotics projects are hindered by problems unrelated to the research objective, such as software or hardware integration issues. This observation system eliminates integration issues by requiring only slight mechanical modification of the robot under observation. One could glue the motion observer to a robot and be done with platform integration. Since most robotic platforms have been designed to allow easy mounting of extra hardware, the mechanical installation requirement of the observation system is not a significant problem for most robots.

In addition to mitigating setup and integration issues, the robot observation system allows detailed data to be easily collected, saved, distributed and replayed for algorithm evaluation. Data collection and distribution is handled by free and open-source software such as ROS, `sshfs`, the `pandas` library for Python, and a script for converting `rosbag` log files to .csv files [62].

3.1.1 Design Constraint

This project was subjected to an unusual constraint that significantly influenced the design of the prototyping system: The author had to design and build a robotic submarine and then deploy it in Antarctica as part of NASA PSTAR grant NNX16AL07G, PI B.E. Schmidt [63, 64]. The author's work on the NASA PSTAR project was completely independent of the work reported in this thesis and vice versa.

Aside from being an activity that consumed the author's time, working in Antarctica motivated a design for a motion observation system that did not rely on Internet access. Internet in Antarctica is intermittent, slow, and only accessible on specially screened computers. Robots were forbidden from connecting to the McMurdo Station network. The Antarctica constraint also required the system to be portable enough to be brought through and potentially operated within airports, hotels and other foreign environments without hassle. Extreme portability and independence from Internet access allowed smooth operation and collection of data in exotic locations: in hotels, airports, in a variety of buildings in

Antarctica, and on a military C-130 transport aircraft as shown in figure 3.1.

Figure 3.1: Author with robot

3.2 Robot Motion Observer Components

3.2.1 Platform Under Observation

Two robotic platforms were chosen to be experimental platforms with separate motivations. A Turtlebot 3 was chosen for its portability and the fact that it can be completely controlled. A Roomba 690 was chosen because it has push-button-ready autonomy. Both were chosen for ease of modification.

ROBOTIS Turtlebot 3

The Turtlebot 3, a significantly more compact and open-source version of its predecessor, the Turtlebot 2 or Kobuki Turtlebot, is similar in size to a 2-liter bottle of soda. Figure 3.2 shows the Turtlebot 3 relative to bricks and a can of soda for a sense of scale. Because of its small size, it is easily portable. The author was able to easily fit this robot in a 25-liter backpack (REI Co-op Trail 25 Pack) used as an airplane carry-on, along with a 15” aspect ratio laptop, a large DSLR camera and three days worth of clothing and toiletries. This allowed the author to set up and conduct motion-observation experiments in under 30 minutes in

arbitrary locations such as airports, hotel rooms, laboratories and even on a military C-130 transport aircraft flying to New Zealand from Antarctica. As a result, the author has a set of data representing similar robot operation in wildly varying environments. The platform is easily duplicated, facilitating simultaneous data collection in many different environments or on many different robots. The Turtlebot 3 was also chosen as a research platform because every piece of it can be easily controlled and modified at its most fundamental level. This eased troubleshooting and the addition of new components such as an IMU, remote controller receiver, and functions such as the ability to serve as a wireless access point and file transfer server.

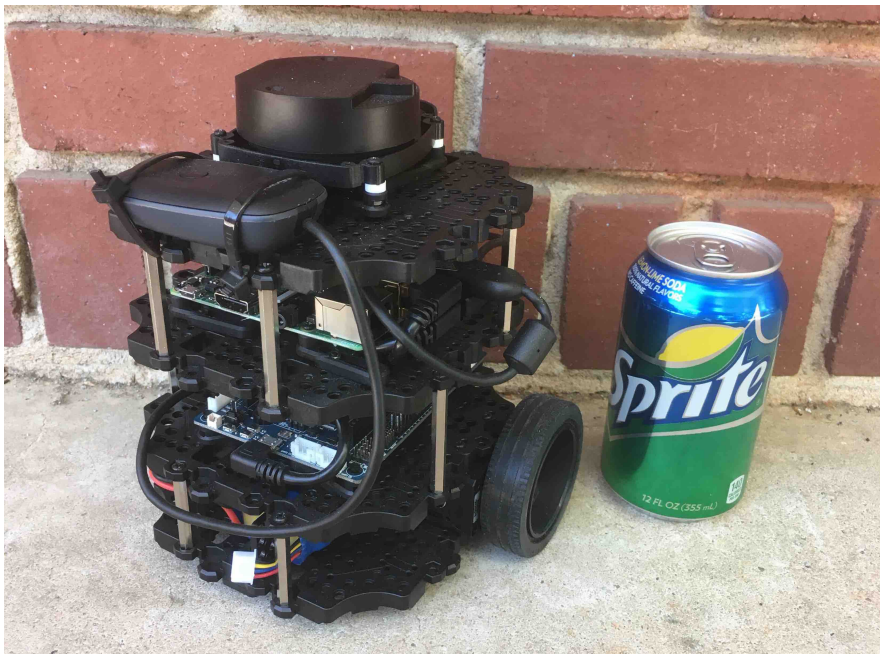


Figure 3.2: Turtlebot 3 with scale reference

iRobot Roomba 690

After experimenting with the Turtlebot 3, the author decided to purchase an iRobot Roomba 690. The instrumented Roomba is shown in Figure 3.3. There were three major motivating factors for this decision. First, the Roomba is a tested and reliable autonomous robot, and the user needs only push a button to start its traverse through a room. The Turtlebot 3 lacks

collison sensors, and requires extra effort to provide robust autonomy. And although it does not take significant effort to make a Turtlebot 3 autonomously traverse a room, a Roomba requires trivial effort to set up and cleans the room as it traverses, whereas the Turtlebot 3 does not.

Second, having a conspicuously clean apartment as a result of data collection appealed to the author.

Lastly, iRobot provides an "Open Interface Specification" that allows users to monitor and modify robot behavior through a serial port on the top of the robot [65]. Although it does not allow the same granularity of control as the Turtlebot, the ability to manually control the robot if needed was still considered a useful feature. Furthermore, the robot motivated the design of the independent motion observation system to sidestep the need to interface with the robot, and to produce measurements comparable with the Turtlebot. A brief overview of the ease of implementing the motion observation platform is provided in Section 3.2.4, in the subsection titled "Notes on scalability".

Figure 3.3: iRobot Roomba 690 with scale reference. IMU visible atop robot.

3.2.2 Hardware Peripherals

Motion Sensors

A system for observing robot motion must have provisions for taking robust and detailed measurements of robot motion. The observation system must allow easy addition and integration of many different sensors. Common ways to sense motions on robots include accelerometers, which measure linear acceleration, and gyroscopes, which measure angular acceleration. Both measurements are derived from the displacement of micro-electro-mechanical (MEMS) proof masses and are, therefore, direct measurements of the forces applied to them. Figure 3.4 shows that a number of motion sensors can be added to the platform: this is a feature of a Phidgets modular hardware system. An example of a multi-node Phidget sensor network is shown in Figure 3.7(a).

The motion sensor used in this research was a PhidgetSpatial Precision 3/3/3 High Resolution IMU [66]. It was chosen because it has a high resolution (73.6 μg), bandwidth (497 Hz), and sample rate (up to 250 Hz), and because its manufacturer provides a well-documented ROS package and API. The reliability and availability of this IMU facilitates duplication of the prototyping system, which the researcher accomplished in a less than a day, mentioned in Section 3.2.1.

Power Supply

To be independent of the robotic platform under observation, the motion observer must include its own power system. Providing this power is simple since the Raspberry Pi draws from a USB-micro port at wattages supported by many off-the-shelf cell phone chargers and portable batteries. A short power cable was used to reduce the voltage drop between the power supply and the battery. A 10,000 mAh portable charging battery (Anker PowerCore 10000) was used to power the Raspberry Pi mounted on the iRobot Roomba used in this research. Measurements of battery percentage with respect to time indicate that it can

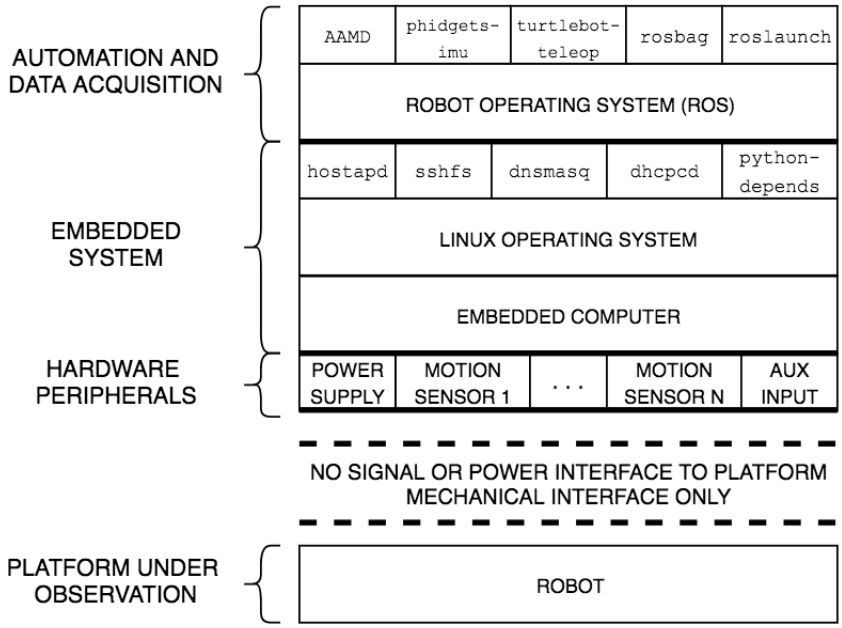


Figure 3.4: Robot-motion-observation platform-component stack

support at least 10 hours of continuous operation given the setup described in this thesis.

Auxiliary Input

The design of the motion observer allows a multitude of hardware peripherals to be connected. The author used a Microsoft Xbox 360 Wireless Receiver to connect to a wireless Xbox 360 controller. The controller inputs were used to generate time-stamped labels indicating when abnormal motion was about to be introduced.

3.2.3 Embedded System

Embedded Computer

The purpose of the embedded computer in Figure 3.4 is to run the Robot Operating System (ROS), interface with sensors, and provide interfaces for data transfer to and control input from an external computer. A Raspberry Pi 3 Model B was chosen to satisfy this purpose. The Raspberry Pi is a low-cost embedded computing platform that has a large community of users, and is compatible with many capable sensors and services. It is also the main

embedded computer in many low-cost or hobbyist robotics platforms, such as the Turtlebot 3, so it is already compatible with many robotics platforms. The Raspberry Pi 3 can also act as a wireless access point, which removes the need to connect it to a local network to connect to it wirelessly. Networking with the Raspberry Pi 3 is briefly discussed in the *Embedded Software Peripherals* section.

Linux Operating System

Linux Ubuntu MATE 15.10 was chosen as the operating system to run on the Raspberry Pi 3 due to its wide userbase and its compatibility with the Robot Operating System (ROS). Ubuntu MATE is also compatible with many software tools that the motion observer relied on, which are discussed in Section 3.2.3.

Embedded Software Peripherals

`hostapd`, a user space daemon, was used to implement a wireless access point with the Raspberry Pi 3 B. `dnsmasq` and `dhcpcd` were used to configure the wireless interface and implement and manage DHCP services. This allowed the control computer, discussed in Section 3.3, to connect to the Raspberry Pi and transfer files without an Internet connection. File transfer was managed with `sshfs`, which allows drives on a network to be mounted to a system over a normal `ssh` connection.

3.2.4 Automation and Data Acquisition

Robot Operating System (ROS)

The Robot Operating System, developed by Willow Garage and supported by the Open Source Robotics Foundation, is an open-source software package that emulates the utilities provided by an operating system, but with special provisions for robots [67, 68]. It is not an actual operating system. Software packages for ROS typically implement programs that can publish or subscribe to data at fixed rates and with standardized message formats.

These programs are called nodes. ROS was chosen for use in the observer because it has many automation tools since it was designed to support the development of autonomous systems. For example, the `rosbag` tool creates time-stamped, replayable recordings of specific streams of data called bagfiles, and it can be used to replay those recordings on almost any other system that runs ROS. The `roslaunch` tool can be used to automatically launch ROS nodes, configure them, and start recording their outputs.

ROS Packages and Nodes

The ROS nodes used on the motion observer were `phidgets-imu` and `turtlebot3_teleop_key`. Occasionally, a `joy_node` node from the `joy` joystick parsing package was used in lieu of the `turtlebot3_teleop_key` node to sense user input. User input was used to label instances of abnormal motions for reference during analysis. The `phidgets-imu` node was configured to publish linear acceleration and angular velocity data at 125 Hz. A `roslaunch` script was used to automatically launch and configure both nodes and a `rosbag` subscriber. The `rosbag` subscriber recorded the `/cmd_vel` and `/imu/data_raw` topics to a file known as a bagfile. A graphical representation of the ROS nodes running on a Turtlebot 3 and publishing data to separate topics is shown in Figure 3.5. The abnormal motion detection algorithm was implemented as a node in a ROS package called `aamd`.

Note on scalability

The robot motion observer described in this thesis can be rapidly duplicated and easily expanded. To illustrate, the author configured the platform on the Roomba and began collecting data with it the day it was received. A program called Etcher by resin.io was used to generate an image of the Raspberry Pi on the Turtlebot 3 and copy it onto another Raspberry Pi, essentially creating a complete clone of the Turtlebot 3 device. Next, a single `git clone` command was used to load the software necessary for data collection

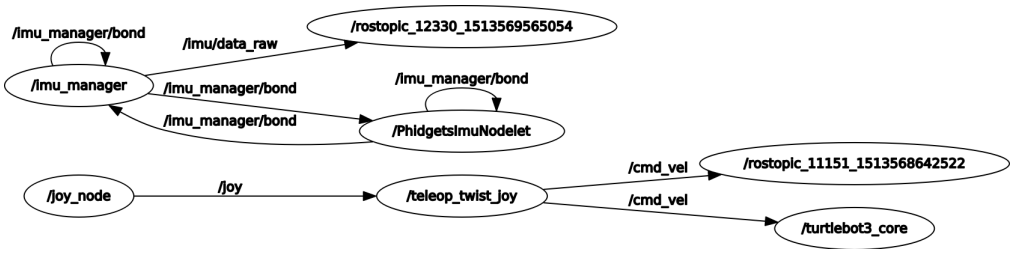


Figure 3.5: `phidgets-imu` and `joy_node` nodes running on a Turtlebot 3

and abnormal motion detection. Securely adhering the Raspberry Pi, external battery, and Phidgets IMU to the Roomba was the most time consuming part of platform setup.

This motion observer allows rapid scaling of the experimental platform. The complete motion observer and robotic platform in action can be seen in Figure 3.6. All of the physical components needed for an implementation can be grouped into one or two online orders available for two-day standard shipping for less than $200, the majority of the cost being the overly-precise IMU, which cost $140 at the time of writing. Furthermore, once one platform is built, its software can be immediately and automatically loaded onto many other Internet-connected motion observers. This is possible with a service like the one provided by Resin.io [69], which allows maintenance, version control and deployment of Linux containers across many Internet-connected computers.

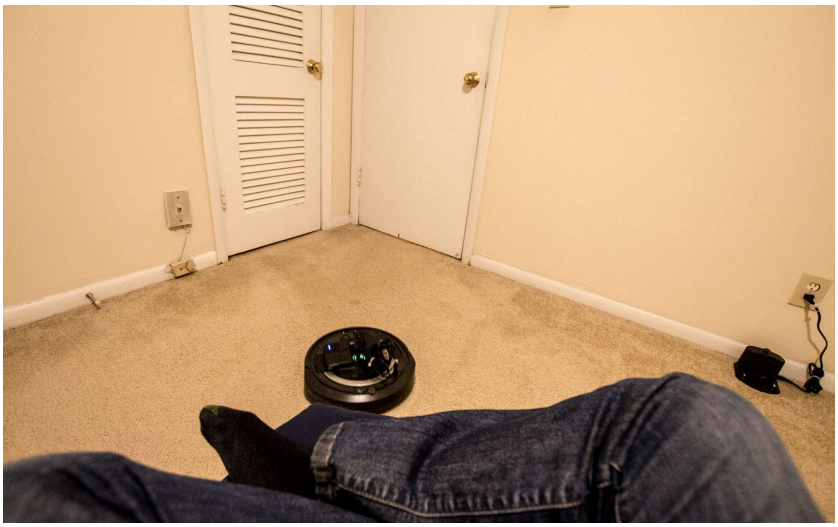


Figure 3.6: Collecting normal-motion data with a Roomba.

3.3 Robot Motion Data Analysis

A dedicated computer, the author's laptop, was used to analyze the data collected by the robot motion observer. The author designed a contained environment for data processing that allowed the entire algorithm analysis and development process to be carried out on any computer that could meet a relatively simple dependency requirement: the ability to run ROS natively or in a virtual machine. The effort to develop a contained environment was motivated by the author's uncertainty in the computing power required to process many hours worth of dense motion data in a reasonable amount of time. Thus, the system was designed to allow deployment to a cloud-based computing platform where arbitrary computing power and memory can simply be rented. This design also facilitates scaling and automation of the overall algorithm development system by allowing parallelization of algorithm-evaluation platforms.

The software written to implement the automatic abnormal motion detection algorithm, to evaluate its performance, and to collect motion data from the robot observer was written entirely in Python. The abnormal motion detection code is stored in a private Git repository as a ROS package, and the analysis and data collection code is stored in a separate Git repository. Each dataset is stored in a directory specific to the robot platform it was collected on. Figure 3.10 shows the structure of the dataset directories.

The motion data analysis environment can be broken into two main components: the development environment and the simulation environment. A flowchart showing the process for conducting an experiment in robot motion analysis is shown in Figure 3.9.

3.3.1 Development Environment

The important aspects of the development environment are the software tools used to perform the software development and algorithm analysis. Any computer or operating system that can satisfy the dependencies of the software tools shown in the Development Environ-

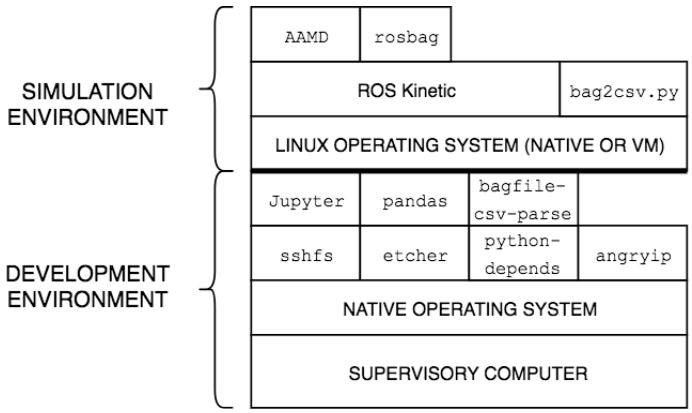


Figure 3.7: Motion-observation-algorithm evaluation stack

ment section of Figure 3.7 will suffice as a development environment. Each software tool is recommended, but any tool that can perform an equivalent or superior function can be used. The tools and their functions are:

- `Jupyter`: Text editor for interactively writing and running Python code.
- `sshfs`: File transfer over a normal `ssh` connection.
- `etcher`: Duplication of operating system images stored on SD cards.
- `python-depends`: Python dependencies of `HDBSCAN`, `Jupyter`, `pandas` and any other software tools.
- `angryip`: Scan IP addresses in local networks. Used to find Raspberry Pis.
- `pandas`: Python Data Analysis Library [70, 71]
- `bagfile-csv-parse`: A script for automatically reformatting and organizing .csv files generated from bagfiles.

3.3.2 Simulation Environment

The author used Parallels Desktop (Parallels, Inc.) to run a Ubuntu 16.04 virtual machine on Apple Macbook Pro running native OSX. This virtual machine allowed the motions

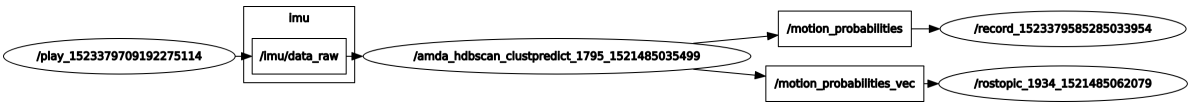


Figure 3.8: Evaluating algorithm performance with `rosbag` playback

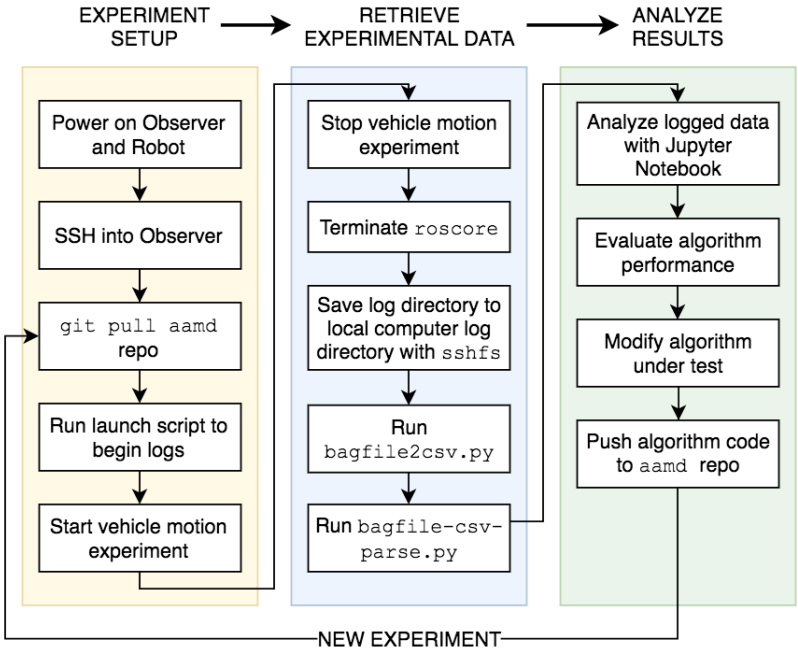


Figure 3.9: Robot motion-analysis experiment process

recorded by the robot observer to be replayed with the `rosbag` utility of ROS, recreating the exact signals that a motion observation algorithm would see on an independent observation system. A graph showing `rosbag` playback can be seen in Figure 3.8. This allowed iterations of the algorithm under development to be evaluated without operating the robot, and it allowed experiments to be automated and run as a background task on the author's laptop.

3.3.3 Robot Observation Process

The tools and systems described in sections 3.3.1 and 3.3.2 supported the experimentation process shown in Figure 3.9.

3.3.4 Experiment Setup

To collect data from a robot, the robot observer is first powered on, and then a connection is established by `ssh`-ing into the embedded computer. The latest version of the motion-analysis algorithm is loaded onto the embedded computer by pulling code from the `aamd` repo, or, in the absence of an Internet connection, from a local repository. Next, a `roslaunch` script is run to initialize ROS and all the nodes necessary for the particular experiment. The script creates a new directory and begins to record data in that directory.

Finally, the vehicle motion experiment is started. In the case of the Roomba, the author sealed it in various rooms in his apartment and initiated jobs with the iRobot HOME app. A normal-motion training dataset was collected simply by letting the robot do its job, taking care to remove anything that could significantly interfere with its performance, such as loose cords. In the case of the Turtlebot 3, the author drove the vehicle in repetitive motions: forward, backward, left, right, and slow clockwise and counter-clockwise circles. Abnormal-motion datasets were simply normal-motion datasets in which the author intentionally interfered with the robot and annotated the time of interference for future reference.

3.3.5 Retrieval of Experimental Data

After a motion experiment concludes, all ROS processes are terminated, and the new directory containing the most recent bagfile is copied into a directory of datasets on the computer with an implementation of the simulation and development environments shown in Figure 3.7. Figure 3.10 shows the structure of the dataset directories.

Each directory of datasets features a subdirectory of "cleaned" datasets which is automatically generated and populated with a Python script. The script reads every `rosbag` file in a directory, detects if it has already been processed, and if not, it transcribes the bagfile into a .csv file with a consistent, `pandas`-friendly format. Each transcribed .csv file is written to the subdirectory titled "pd_cleaned_csvs_(epoch timestamp)", where the epoch

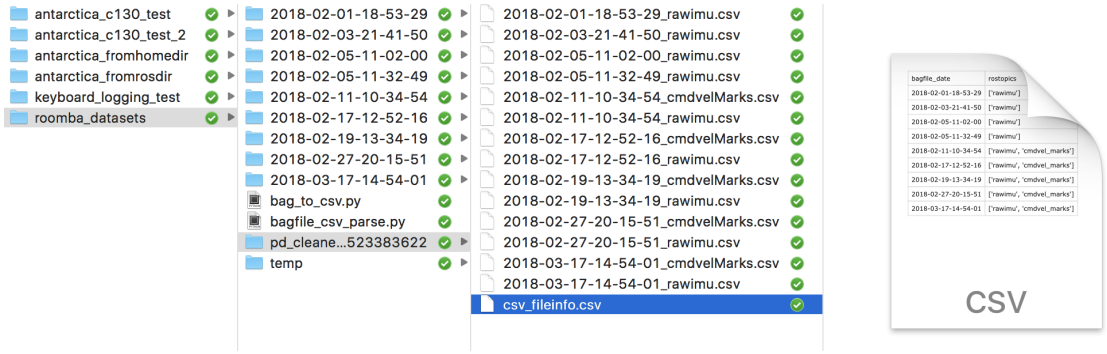


Figure 3.10: Structure of motion dataset directories

timestamp marks the date and time that the directory was created. The rosbag transcription script also creates a .csv file named "csv_fileinfo.csv" in the directory that provides convenient information summarizing the processed bagfiles.

3.3.6 Analysis of Results

Once the experimental data is transcribed into a convenient format, the performance of the algorithm can be evaluated interactively with a Jupyter notebook. Typical experimental data consists of time-stamped acceleration, anomaly-indication, and motion-probability signals. A typical analysis will seek to test a hypothesis regarding the design of the abnormal motion detection algorithm. For example, one hypothesis that was tested during one analysis was "Applying a low-pass filter to the acceleration data will reduce the variance in the motion-probability output given its current implementation". This hypothesis was tested, confirmed, and low-pass filtering was incorporated into the algorithm.

Finally, once the analysis of the experimental data is concluded, the discoveries made during the analysis are used to guide modifications to the abnormal motion detection algorithm. The updated algorithm is then loaded onto the motion-observer, and the process repeats.

3.4 Automatic Abnormal Motion Detection

The goal of automatic abnormal motion detection is to differentiate between "normal" and "abnormal" behavior without expert guidance on a mobile robot. The rapid prototyping system for robot observation described in Section 3.1 was used to collect the data required to develop this algorithm.

To detect abnormal motions, a model representing normal motions must first be built. An algorithm was written to generate these models. It is explained in Section 3.4.1, and a flowchart illustrating the algorithm is shown in Figure 3.11. The abnormal motion detection algorithm uses these models to generate a probabilistic prediction of whether an observed motion is normal or not. This algorithm is described in Section 3.4.2, and a flowchart illustrating how it works is shown in Figure 3.13.

3.4.1 AAMD Model Generation

The abnormal motion detection algorithm developed in this research relies on `HDBSCAN`, a hierarchical density-based clustering algorithm written in Python [54]. HDBSCAN can correctly separate clusters of data that would be difficult for other algorithms to separate such as concentric, overlapping, or U-shaped clusters. HDBSCAN also generates a representation of the discovered clusters in the form of a condensed tree, which allows comparison of the similarity between separate clusters. HDBSCAN only has two hyperparameters that must be tuned: `min_cluster_size` and `min_sample_size`. The effects of tuning these parameters was not examined in this thesis. The algorithm's ability to robustly detect hierarchies of clusters in noisy data motivated its inclusion in the abnormal detection algorithm. Its ability to generate a probabilistic prediction of whether a new data point will fit into any of the existing clusters was the other motivating factor in the selection of HDBSCAN. The abnormal motion detection algorithm relies on this function, named `approximate_predict()`, to calculate the probability that an observed motion (a new

data point) fits into expectations of normal symptomatic motions (existing clusters).

A clustering object serves as the model of normal motion, and the output of `approximate_predict()` is the raw output of the abnormal motion detection algorithm. To be useful, the model must be trained with a dataset of normal motion. Figure 3.11 shows the dataset-generation process. This process had many options to facilitate experimentation with the effects of the training dataset on algorithm performance.

The work presented in this thesis is preliminary, and used the simplest options in every instance to generate a simple proof-of-concept implementation. The goal was to verify that abnormal motion detection could be performed with the independent observation system described in Section 3.1, and that even a minimalist use of HDBSCAN (no features, thresholding or other encoded expert information) could be used to quickly detect at least some abnormal motions.

First, a training dataset is assembled by either importing a single recording of "normal" motion data, or by merging many recordings of motion data together. In this research, x and y-axis acceleration data from a single recording was used as the source data. The dimensionality of the input data was 2x287,564. Next, the data was filtered to remove noise. A fourth-order 3 Hz low-pass Butterworth filter was used to attenuate high-frequency noise. This was important due to the presence of the vacuum and brush motors on the Roomba, which generated a broad spectrum of vibrations. The filter was implemented with `scipy.signal.lfilter`, which caused a negligible phase shift in the signal being processed.

After filtering, the input data can be partitioned to match a partitioning scheme used by the abnormal motion detection algorithm on the robot motion observer. Data may be partitioned by time-windowing, for the purpose of examining features within a fixed amount of time, such as the mean and variance of x-acceleration experienced every second. Similarly, data can be partitioned by number of samples if the sensor is asynchronous and does not publish data at a consistent rate. No partitioning was chosen for the model used in this im-

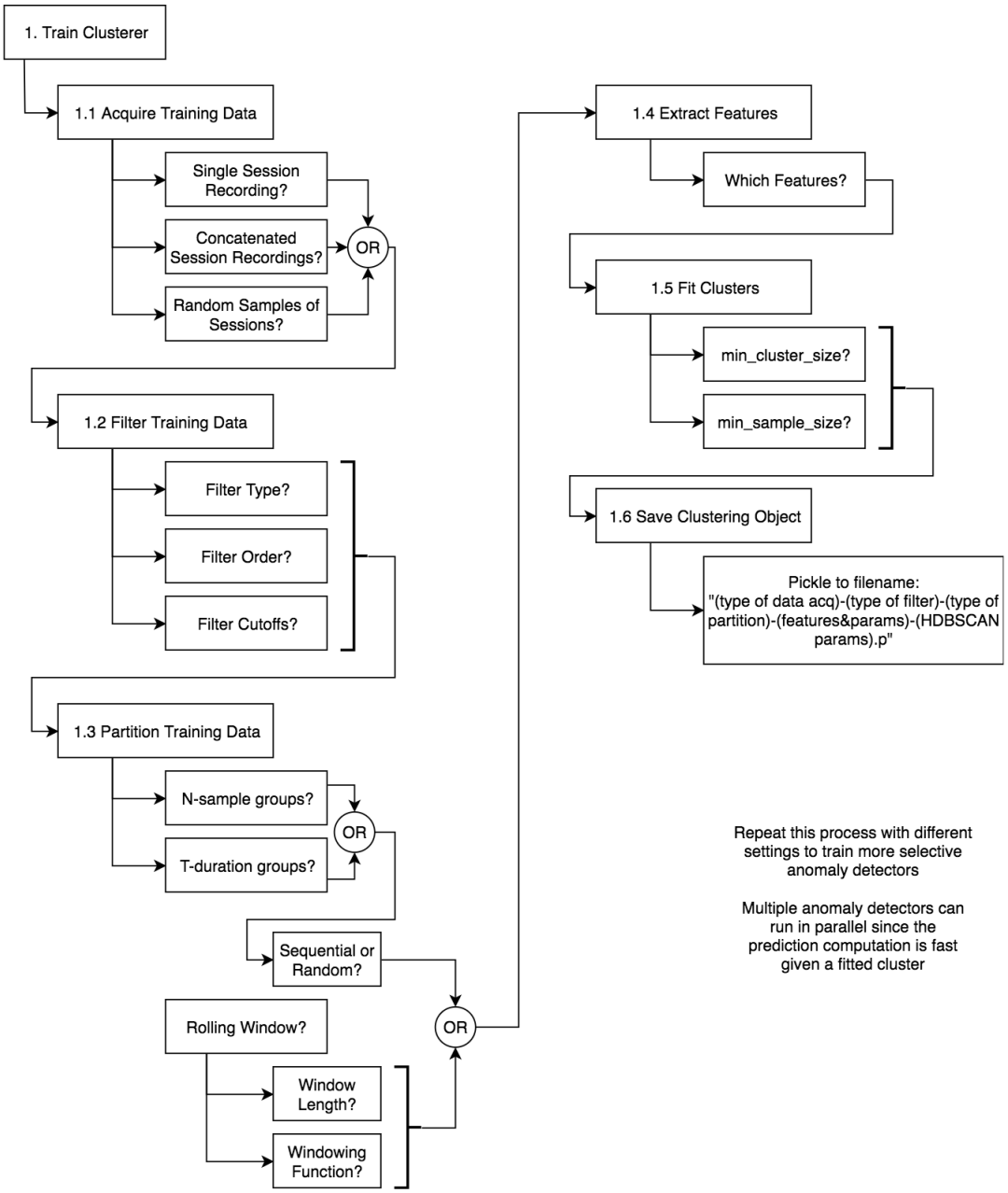


Figure 3.11: Offline training of the abnormal motion detector

```
In [38]:  1 %%time
          2 test = AMDAClusterTrainingLib()
          3 test.train_amda()

Pathway to search for pd-cleaned_csv:
Press enter without entry to search pwd.
OVERRIDE: default search set to '/Users/ddichek/Dropbox (GaTech)/Thesis' for convenience.
Attempting to search /Users/ddichek/Dropbox (GaTech)/Thesis
WARNING: found multiple pd-cleaned-csv directories.
Choosing the last directory found.
Fix your code if you don't like this behavior.
Found pd-cleaned-csv directory:
----------------------
---> /Users/ddichek/Dropbox (GaTech)/Thesis/bags/roomba_datasets/allbags/pd_cleaned_csvs_1523383622
----------------------
Found the following logs:
          bagfile_date                  rostopics
0  2018-02-01-18-53-29                 ['rawimu']
1  2018-02-03-21-41-50                 ['rawimu']
2  2018-02-05-11-02-00                 ['rawimu']
3  2018-02-05-11-32-49                 ['rawimu']
4  2018-02-11-10-34-54  ['rawimu', 'cmdvel_marks']
5  2018-02-17-12-52-16  ['rawimu', 'cmdvel_marks']
6  2018-02-19-13-34-19  ['rawimu', 'cmdvel_marks']
7  2018-02-27-20-15-51  ['rawimu', 'cmdvel_marks']
8  2018-03-17-14-54-01  ['rawimu', 'cmdvel_marks']

Enter log selection: 0
Reading: 2018-02-01-18-53-29_rawimu.csv
       rosbagTimestamp        secs     nsecs       x.1       y.1       z.1  \
0  1517529210078354889  1517529210  54671173 -0.003867  0.001055  0.001522
1  1517529210085325989  1517529210  62671174 -0.002109 -0.001230  0.001405
2  1517529210093296927  1517529210  70671173 -0.002988  0.001406  0.000937
3  1517529210101303020  1517529210  78671173 -0.003515  0.006328  0.000117
4  1517529210109301353  1517529210  86671174 -0.005449  0.005273  0.000351

        x.2       y.2       z.2
0 -0.644419 -0.928026 -9.741330
1 -0.644027 -0.928026 -9.741330
2 -0.644811 -0.924298 -9.741723
3 -0.644027 -0.922434 -9.742115
4 -0.638042 -0.926947 -9.744764
Writing to memory and exiting sloppily.
Saving pickled cluster object as s-lp3o4-xy-500_350.p...
CPU times: user 27.2 s, sys: 6.83 s, total: 34 s
Wall time: 55.1 s
```

Figure 3.12: Interactive generation of a HDBSCAN cluster model

plementation: the entirety of the x and y acceleration experienced during a single normal job was clustered.

The next major step in the algorithm allows a feature-extraction algorithm to be added. An entire other thesis could be written on the design of a feature extraction algorithm to improve clustering results in this application. Since this algorithm is meant as a rudimentary proof of concept, no special features were generated.

Finally, a cluster object is generated with `HDBSCAN`'s `fit()` method. The 2 hyperparameters for HDBSCAN, min_cluster_size and min_samples, were set to 500 and 350, respectively. These values were chosen by iteratively setting the parameters, plotting cluster groupings of the input data, and adjusting one parameter or the other based on the resulting clusters and the advice of the HDBSCAN documentation. Once the cluster object is generated, it is saved for use on the robot with the Python `pickle` library. The saved cluster object is given a filename indicative of the parameters used to train it.

A `Jupyter` notebook was used to implement this model-generation algorithm. This

allowed the author to use simple text commands to specify the parameters for a clustering object to be generated. A screenshot of the rudimentary interactive process of generating a model is shown in Figure 3.12. The execution time of a single complete run of this cluster-model-generation algorithm on a single 20 MB 2x287,564 dataset was measured with `Jupyter`'s `%%time` feature and can be seen at the bottom of Figure 3.12 (55 seconds).

3.4.2 AAMD Algorithm Implementation

Figure 3.13 shows a diagram of the abnormal motion detection algorithm. This algorithm is implemented as a ROS node, "`amda_hdbscan_clustpredict`" which can be seen in Figure 3.8. It publishes a vector of motion probabilities.

Some of the components shown in the diagram were not implemented in this research, but they would be required for a truly complete abnormal motion detection algorithm. This research focused on a new clustering algorithm, HDBSCAN, for abnormal motion detection.

The first few steps of the abnormal motion detection algorithm can be thought of as an incremental version of the motion model generation process. First, data is acquired by the sensor and then partitioned, if the features to be extracted call for specific sample window times or numbers of samples. This research implemented no partitioning function, although the structure of the buffer (seen after Motion Sensor 1 in Figure 3.13) allows either continuous streaming of sensor values or for partitioning into windows. This feature was implemented with Python `deque` data structures.

The same filter used to train the motion model (3 Hz, 4th order Butterworth) was implemented in the abnormal motion detection algorithm. No features were extracted. The filtered data was partitioned into groups of 100 samples.

After the filtering step is the core of the abnormal motion detection algortihm: the prediction step. When the "`amda_hdbscan_clustpredict`" node is started, a Python-

pickled `HDBSCAN` cluster object is loaded, and its `approximate_predict()` method is called to calculate the probability that the recent motions were "normal motions". This is the extent to which abnormal motion was implemented in this research.

The rest of the process flow - the "Window-probability comparator" and "Anomaly annunciator" - are functional elements that were performed by the author manually. The job of the window-probability comparator is to provide an estimate of how likely the current observations are given previous observations. In other words, it conditions the current HDBSCAN-generated probability on prior HDBSCAN probabilities and/or other probabilities indicative of abnormal motion. The anomaly annunciation function could be implemented with a simple thresholding function to generate a boolean output, but this would only serve to help quickly evaluate performance. Specifically, one could use it to compare the timestamps of indicated anomalies with known anomalies. This was done manually.

Figure 3.8 also shows how the model-generation process of Section 3.4.1 fits into the abnormal motion detection algorithm. An idea for how the algorithm could be made online is also shown.

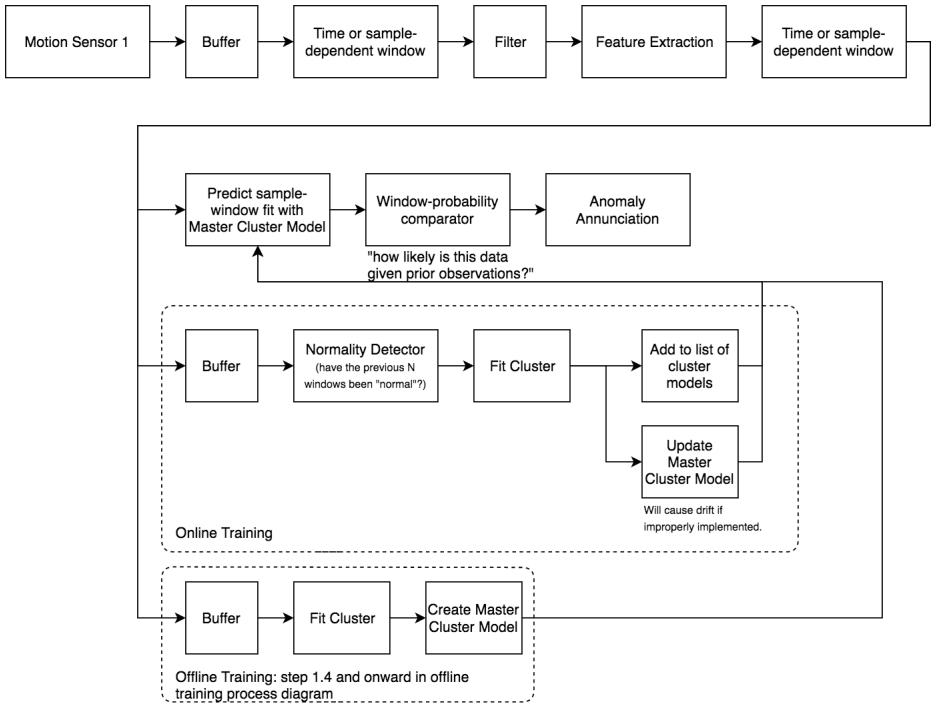


Figure 3.13: Algorithm for automatic abnormal motion detection

CHAPTER 4

RESULTS

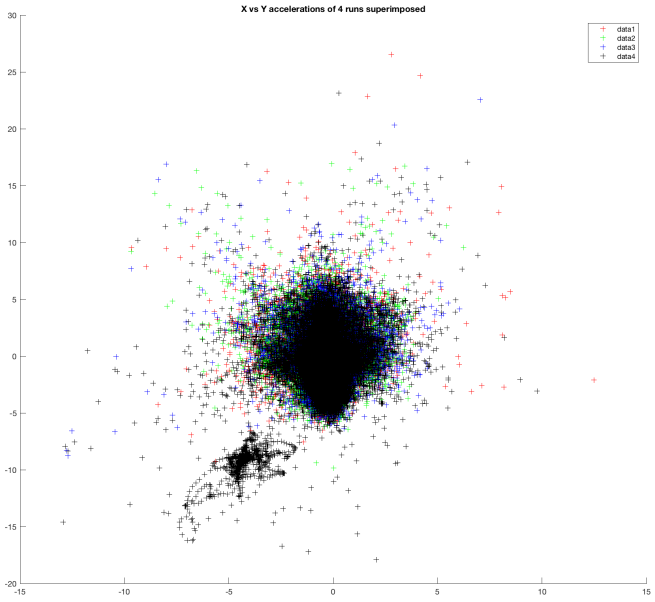
4.1 Symptomatic Motions in Robotic Platforms

IMU data collected from the Roomba test platform is shown in Figure 4.1. All figures show high concentrations of several combinations of accelerations. This suggests the presence of symptomatic motions. These symptomatic motions were correctly clustered by the implementation of HDBSCAN described in Section 3.4.1.

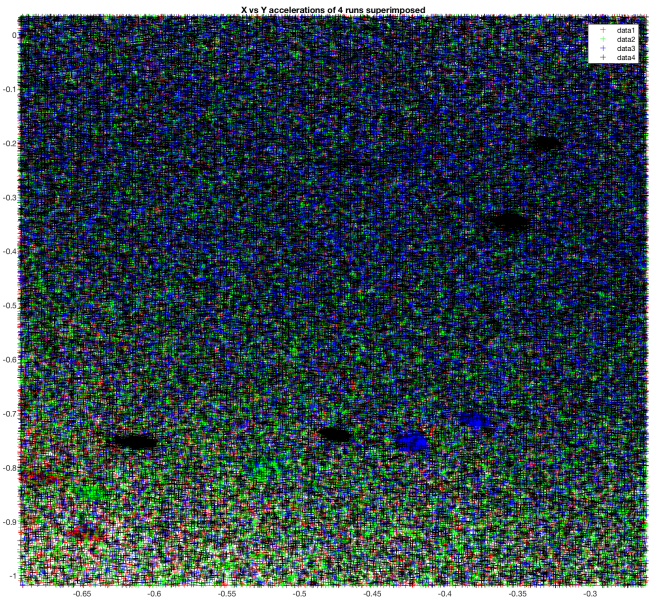
Figure 4.2a shows color-coded clustered points from the first of the four datasets overlaid in Figure 4.1. Figure 4.2a is also a birds-eye-view of Figure 4.3a. There are three clusters identified in Figure 4.2a, represented by red, green and blue points. Grey points are classified as outliers. Note that the symptomatic motions have been correctly identified as separate clusters despite being buried in noisy acceleration data. They can be seen emerging in Figure 4.2b, and more clearly in 4.2c. The fact that there are two distinct clusters of accelerations suggests that they cannot both be due to the robot reading the same acceleration offset while sitting still. Further evidence to support the idea that these clusters reflect symptomatic motions is the fact that the green cluster is significantly more elongated than the blue cluster, showing a greater variance in acceleration in the x-direction for a given value of acceleration in the y-direction.

4.2 Detecting Abnormal Motions with Data Visualization

Figures 4.2 - 4.4 show histograms of the x and y accelerations during a testing run. Figures 4.3a - 4.3c show motion during normal operation. Note the similar shape. Also note the consistent asymmetry in the distributions. Careful examination will reveal a few points floating above the majority of the distribution, suggesting particular combinations of ac-

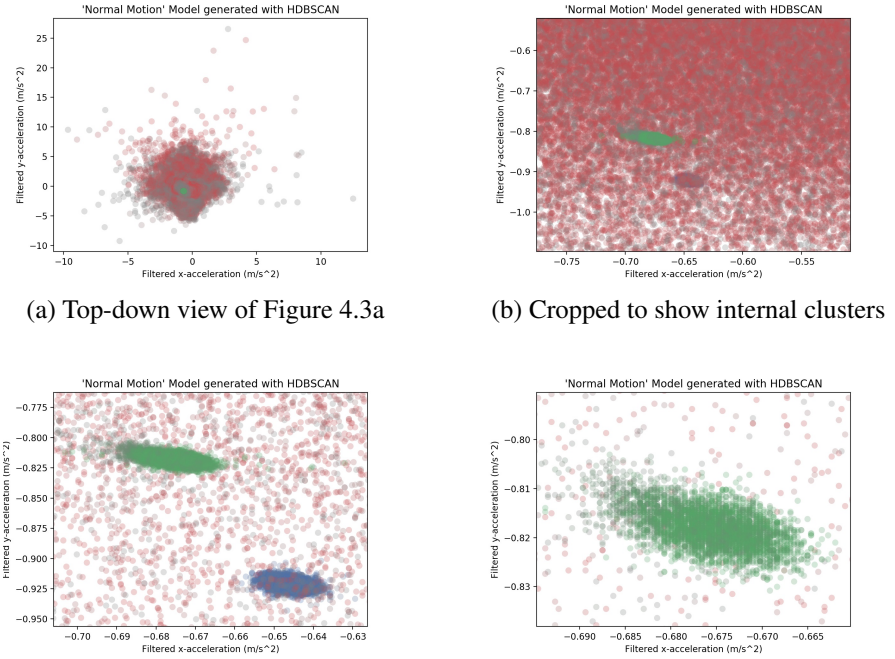


(a) Overlaid X-Y acceleration histogram from 3 normal runs and 1 abnormal run



(b) Closeup of (a) revealing symptomatic motions

Figure 4.1: Histograms showing symptomatic motions



(a) Top-down view of Figure 4.3a

(b) Cropped to show internal clusters

(c) Symptomatic motions clustered

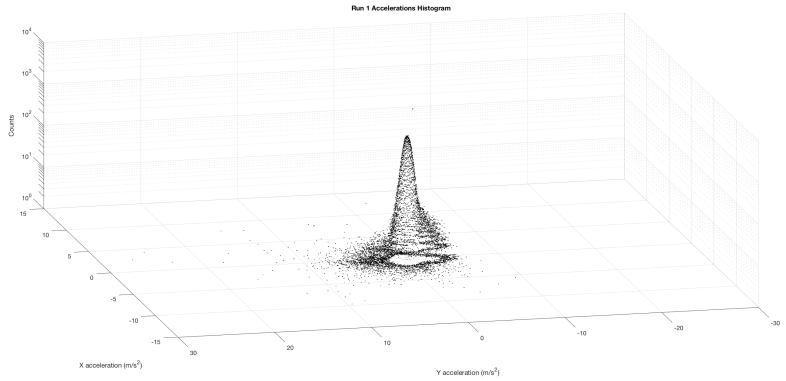
(d) Close-up of clustered accelerations

Figure 4.2: HDBSCAN revealing symptomatic motions

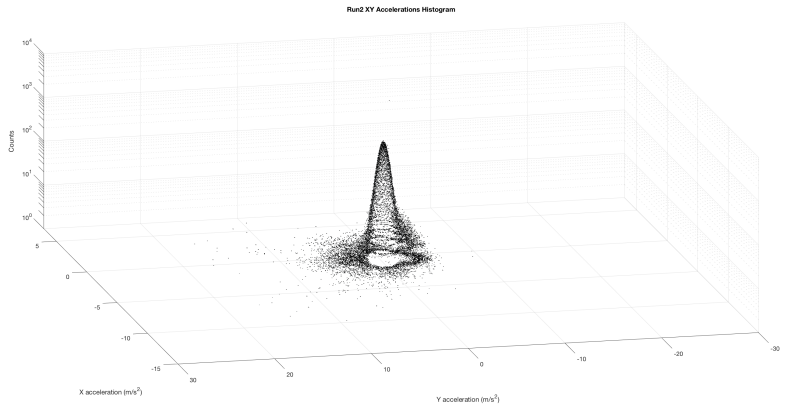
celerations that are experienced an order of magnitude more frequently than their nearest neighbors. Figure 4.4 shows a run in which the robot was lifted up and moved. The abnormal motions in 4.4 are clearly visible. They can also be seen relative to the combined accelerations of every other run in Figure 4.1a.

4.3 AAMD Algorithm Output Given Uneventful Datasets

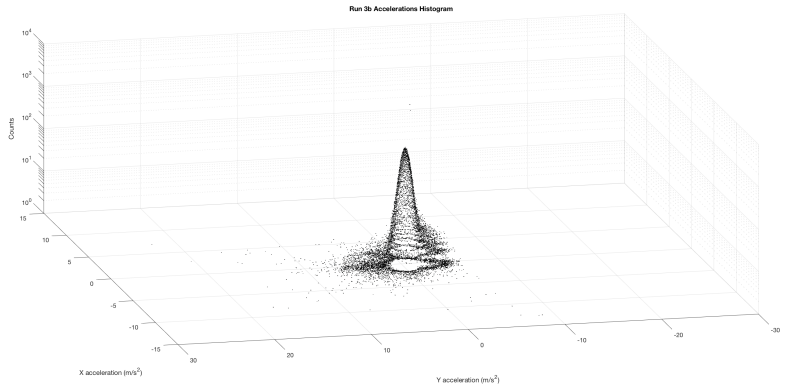
The AAMD algorithm correctly indicated no abnormalities when tasked with analyzing previously-unseen datasets without abnormalities. This can be seen in Figure 4.5. The blue "plus" symbols indicate an HDBSCAN-computed probability. Note that most of the motions are considered "probable" by HDBSCAN. Compare this figure with figures 4.7 and 4.6. Also note that the momentary spike about 1,500 data points into the dataset is not classified as an abnormality despite being unlike anything else in the dataset. This is correct: spikes like these happen rarely, but they are normal motions. They typically occur



(a) Semilog acceleration histogram - first run, normal operation



(b) Semilog acceleration histogram - second run, normal operation



(c) Semilog acceleration histogram - third run, normal operation

Figure 4.3: 3D semilog acceleration histograms showing normal motions. Note similarity.

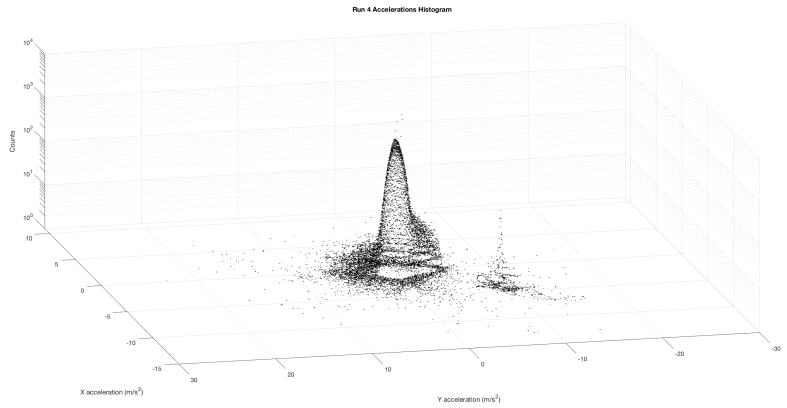


Figure 4.4: 3D semilog acceleration histogram - fourth run, human interference

when the Roomba gains momentum and collides with a rigid an object without glancing off it obliquely.

4.4 AAMD Algorithm Output Given Datasets With Abnormalities

The AAMD correctly identified abnormalities in previously-unseen runs. Figure 4.6 is a contrast to Figure 4.5: both plots show acceleration (top) and probability (bottom) with respect to time. Figure 4.6 shows the output of the algorithm given a dataset with anomalies, and Figure 4.5 shows the output of the algorithm given a dataset without anomalies. Four clear dips in motion-probability are seen in the lower plot of Figure 4.6, corresponding to the times of the four disturbances in this dataset.

A close-up of an anomaly detected in a separate run is shown in Figure 4.7. The red "plus" symbols represent probability, and the blue line represents the acceleration experienced. Note how most of the probability estimates drop to zero when the acceleration signal suggests a significant deviation from "normal" operation. In this instance, the robot was lifted briefly, set down, and then restarted.

Note that in the period before the robot was restarted, in which the acceleration signal is nearly zero, the HDBSCAN probability follows seemingly discrete curves. This is likely due to the highly similar but not completely identical forces experienced by the robot while

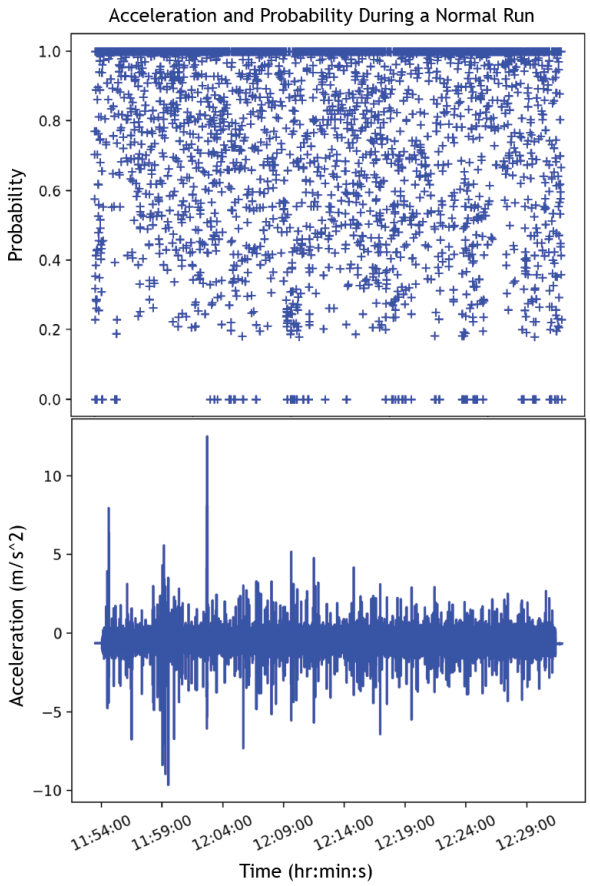


Figure 4.5: No anomalies detected in unseen dataset of normal motion

it is at rest on two separate occasions. The robot was trained to recognize a model of "stationary" on one dataset, and it is ranking the stationary period as consistently similar to something it has experienced before.

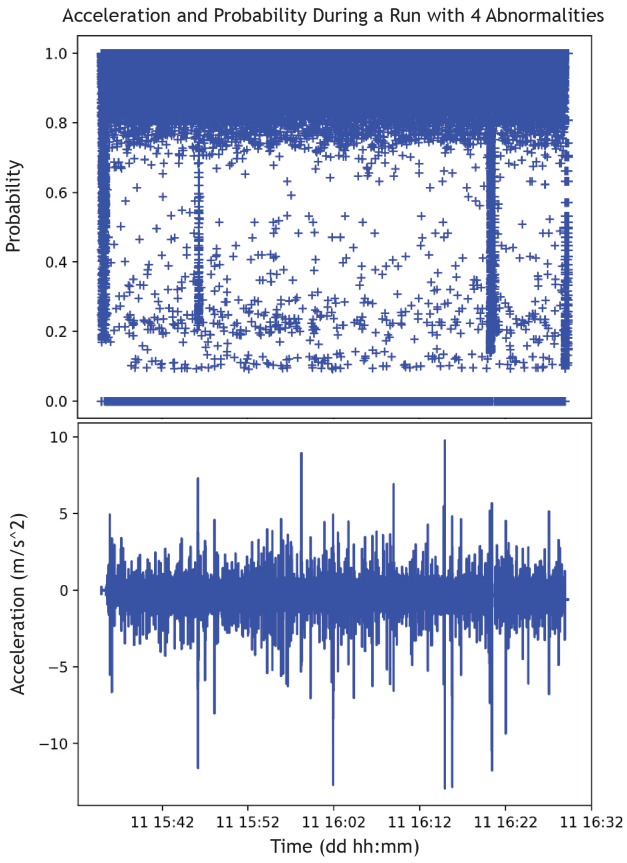


Figure 4.6: Anomalies detected in previously-unseen dataset with abnormal motion

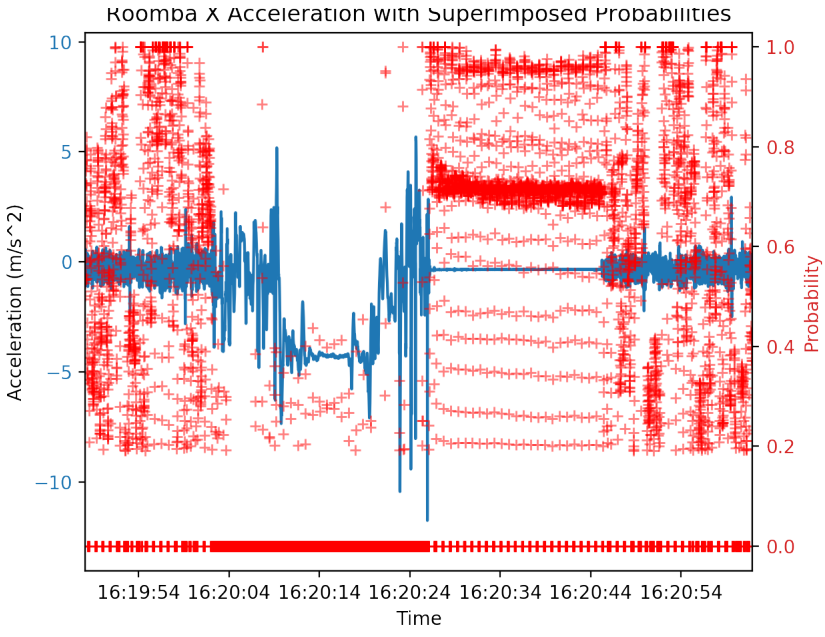


Figure 4.7: Motion probability (red, bounded 0 to 1) versus acceleration (blue)

CHAPTER 5
DISCUSSION

This thesis presents preliminary results from a novel system for automatically detecting abnormal motion in mobile robots. It demonstrated that a model of normal motion can be built and then used as a basis for identifying motions that a human observer would consider abnormal. The benefits of the hierarchical density-based clustering algorithm HDBSCAN were shown to be its ability to correctly separate overlapping clusters of data and its ability to quickly calculate a probability of a motion being "normal" given a trained cluster model of normal motion. This functionality may help reduce robot reliance on humans for anomaly detection.

However, this work is preliminary. The intent of this research was to conduct a proof-of-concept study to determine the feasibility and challenges of implementing abnormal motion detection. This work presents shows that a bare-minimum implementation, which lacked features, thresholding functions or any other form of data pre-processing: the algorithm operated on nearly raw acceleration data. Results are not presented qualitatively, and the findings are not thoroughly compared to prior work. The promising results of this rudimentary implementation justifies further study into feature engineering, the impact of different sensors, the "resolution" of abnormal motion detection as well as how to define it, examining instances of false positives/negatives, methods for qualitatively evaluating effectiveness, and more. The success of the independent motion observation platform in allowing rapid collection of useful data also merits further study into how it can be made into a standardized tool for other researchers.

It is important to note the limitations of the results reported in this research. Although symptomatic motions can be used to characterize robot behavior, they rely on the observation system's perception of robot motion. Care must be taken to select sensors that will

have the resolution and dynamic range to represent motions accurately enough for them to be separated by the clustering algorithm. Furthermore, generating features that are independent of each other, that correlate strongly with motion, and are sparse enough to ward off the curse of dimensionality is an act of high wizardry. Feature engineering is a field of study all of its own. Nevertheless, feature engineering is a problem that should be vigorously attacked because it is the key to getting machine learning algorithms to work well.

Another serious limit to functionality in the current implementation is due to the nature of the probabilistic outputs of HDBSCAN. Figure 4.7 illustrates a mysterious behavior: HDBSCAN never outputs a probability between 0 and .2. This is not a truly probabilistic output. Furthermore, a probabilistic output that includes 0 or 1 in its output distorts or destroys probabilistic distributions that it is used to condition: computing a joint probability between a prior probability and a probability of 0 returns a probability of 0, which is not helpful. Further work will be required to examine the implementation of the probabilistic prediction function of HDBSCAN and make it amenable to contemporary probabilistic methods for robotics.

Ideally, the algorithm presented could be evaluated quantitatively and automatically. Figure 5.1 presents a possible implementation. There are many more complicated problems that must be overcome to implement a truly online, self-evaluating abnormal motion detector, but it seems feasible provided a herculean effort.

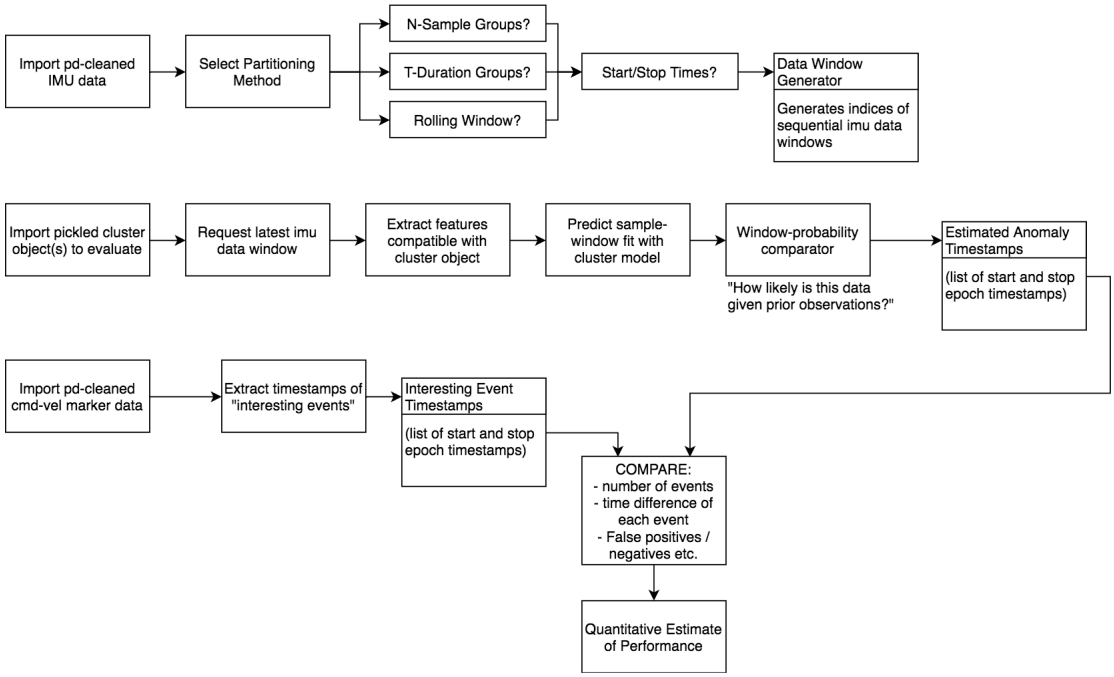


Figure 5.1: Potential automatic algorithm evaluation method

CHAPTER 6
CONCLUSION

A novel method for automatically detecting abnormal motions was implemented and its performance was examined. The preliminary results show that hierarchical density-based clustering has a promising ability to extract complicated structures from motion data. These structures form a model of "normal motion" that can be used to determine whether an observed motion is normal or not. A convenient name was termed for these structures: "symptomatic motions". The term stems from the fact that robot construction and control biases robot motion, and a robot that repeats a task frequently will experience symptoms of these motion biases. Symptomatic motions were clearly identified visually and by the HDBSCAN hierarchical density-based clustering algorithm.

This research also presented an independent system for robot motion observation. The self-contained nature of the observation system allows rapid collection of experimental data by minimizing integration effort and by providing many tools for automating experiments and data collection. This system was used to collect all of the data used in this research.

Future work aims to solidify the work presented in this thesis as a set of open source tools for researchers interested in detecting abnormal motions in robots. The abnormal motion detection algorithm described in this thesis is currently implemented as a ROS package, which is a software package that many roboticists can easily incorporate in their existing systems. The software in the package will be refactored to conform to the ROS package and code style guides. Once this is done, the package will be released as open-source software.

Ultimately, the author hopes to examine the incorporation of many more types of motion sensor to refine the ability of the abnormal motion detector. Automated and parallelized testing is another envisioned feature to expand upon in future work.

As robots continue to invade daily life, it is more important for them to be able to behave themselves without requiring human supervision. Current methods for detecting and reacting to abnormal conditions - even some of the most robust and widely-relied-upon - still rely on humans to identify abnormalities and initiate action. Enabling robots to generate their own distinction between "normal" and "abnormal" given a series of observations will help them become more independent of human operators. Robots made more useful and reliable by enabling them to detect abnormal motions could fulfill roles we wish they could today. If given a sensation of discomfort - a sense that something is abnormal, not as it should be - we may be able to trust unsupervised robots.

REFERENCES

[1] P. Abbeel, V. Ganapathi, and A. Y. Ng, "Learning Vehicular Dynamics, with Application to Modeling Helicopters," *Advances in Neural Information Processing Systems (NIPS)*, pp. 1–8, 2005. [Online]. Available: http://books.nips.cc/nips18.html.

[2] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 3223–3230, 2015.

[3] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection," [Online]. Available: https://arxiv.org/pdf/1603.02199.pdf.

[4] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017. [Online]. Available: www.elsevier.com/locate/artint.

[5] *A Compilation of Robots Falling Down at the DARPA Robotics Challenge - YouTube*. [Online]. Available: https://www.youtube.com/watch?v=g0TaYhjpOfo.

[6] J. Bartlett, *Knightscope Delivers Incident Report On DC 'Drowning'*, 2017. [Online]. Available: https://security.world/knightscope-delivers-incident-report-on-dc-drowning/.

[7] M. Liu, *Knightscope issues report on robot incident at Stanford Mall*. [Online]. Available: https://www.stanforddaily.com/2016/07/25/knightscope-issues-report-on-robot-incident-at-stanford-mall/.

[8] M. McFarland, *Tesla's autopilot probed by government after fatal crash kills driver - Jun. 30, 2016*. [Online]. Available: http://money.cnn.com/2016/06/30/technology/tesla-autopilot-death/index.html?iid=EL.

[9] H. Somerville, P. Lienert, and A. Sage, *Uber's use of fewer safety sensors prompts questions after Arizona crash*. [Online]. Available: https://www.reuters.com/article/us-uber-selfdriving-

sensors-insight/ubers-use-of-fewer-safety-sensors-prompts-questions-after-arizona-crash-idUSKBN1H337Q.

[10] K. Hill, *Volvo says horrible 'self-parking car accident' happened because driver didn't have 'pedestrian detection'*. [Online]. Available: https://splinternews.com/volvo-says-horrible-self-parking-car-accident-happened-1793847943.

[11] R. Wert, *Volvo Pedestrian Avoidance Test Fails Spectacularly*. [Online]. Available: https://jalopnik.com/5648126/volvo-pedestrian-avoidance-crash-test-fails-spectacularly.

[12] J. Croft, *Video: How Does A 737 Handle On A Full Stall? — UPRT At Alaska Airlines — Commercial Aviation content from Aviation Week*. [Online]. Available: http://aviationweek.com/commercial-aviation/how-does-boeing-737-handle-full-stall.

[13] F. A. Administration, *Amdt. 23-7, 34 FR 13087, Aug. 13, 1969, as amended by Amdt. 23-45, 58 FR 42159, Aug. 6, 1993; Amdt. 23-50, 61 FR 5191, Feb. 9, 1996*. [Online]. Available: https://www.gpo.gov/fdsys/granule/CFR-2002-title14-vol1/CFR-2002-title14-vol1-sec23-207.

[14] *Aerodynamic Flutter - YouTube*. [Online]. Available: https://www.youtube.com/watch?v=3mJ_CAR52h4\app=desktop.

[15] J. L. Gloss, *Henry Ford and the Auto Assembly Line*. [Online]. Available: https://www.thoughtco.com/henry-ford-and-the-assembly-line-1779201.

[16] CA DMV, *Adopted Regulatory Text for Driverless Testing Regulations*, 2018. [Online]. Available: https://www.dmv.ca.gov/portal/wcm/connect/a6ea01e0-072f-4f93-aa6c-e12b844443cc/DriverlessAV_Adopted_Regulatory_Text.pdf?MOD=AJPERES.

[17] A. Holland Michel, D. Gettinger, and H. Michel, "DRONE YEAR IN REVIEW: 2017," Center for the Study of the Drone, Tech. Rep., 2018. [Online]. Available: http://dronecenter.bard.edu/drone-year-in-re-view-2017/..

[18] Federal Aviation Administration, *FAA Releases Updated Drone Sighting Reports*, 2017. [Online]. Available: https://www.faa.gov/news/updates/?newsId=87565.

[19] D. Etherington, *Waymo orders thousands of Pacificas for 2018 self-driving fleet rollout*, 2018. [Online]. Available: https://techcrunch.com/2018/01/29/waymo-orders-thousands-of-pacificas-for-2018-self-driving-fleet-rollout/.

[20] R. Medford, "Disengagement Report Report on Autonomous Mode Disengagements For Waymo Self-Driving Vehicles in California," 2017. [Online]. Available: https://www.dmv.ca.gov/portal/wcm/connect/42aff875-7ab1-4115-a72a-97f6f24b23cc/Waymofull.pdf?MOD=AJPERES.

[21] A. Boniske, "GE Cruise Autonomous Vehicle Tester Program Annual Disengagement Report," 2017. [Online]. Available: https://www.dmv.ca.gov/portal/wcm/connect/d94d9334-9955-4f97-aae1-5a2c9f10673b/GM_Cruise.pdf?MOD=AJPERES.

[22] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, p. 78, 2012. arXiv: 9605103 [cs].

[23] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 3406–3413, 2016. arXiv: 1509.06825.

[24] S. Ferrari and R. F. Stengel, "Online Adaptive Critic Flight Control," *JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS*, vol. 27, no. 5, 2004. [Online]. Available: http://www.princeton.edu/~stengel/JGCD2004.pdf.

[25] K. Bousmalis and S. Levine, *Closing the Simulation-to-Reality Gap for Deep Robotic Learning*, 2017. [Online]. Available: https://research.googleblog.com/2017/10/closing-simulation-to-reality-gap-for.html.

[26] W. Yu, C. K. Liu, and G. Turk, "Preparing for the Unknown: Learning a Universal Policy with Online System Identification," 2017. arXiv: 1702.02453. [Online]. Available: http://arxiv.org/abs/1702.02453.

[27] H. Soh and Y. Demiris, "Incrementally learning objects by touch: Online discriminative and generative models for tactile-based recognition," *IEEE Transactions on Haptics*, vol. 7, no. 4, pp. 512–525, 2014.

[28] S. P. Chatzis and Y. Demiris, "Echo State Gaussian Process," *IEEE Transactions on Neural Networks*, vol. 22, no. 9, pp. 1435–1445, 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5966349/.

[29] V. Krüger, D. Kragic, A. Ude, and C. Geib, "The meaning of action: a review on action recognition and mapping," *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, 2007. [Online]. Available: http://www.tandfonline.com/doi/pdf/10.1163/156855307782148578?needAccess=true.

[30] W. contributors, *List of datasets for machine learning research — wikipedia, the free encyclopedia*, [Online; accessed 6-July-2017], 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=List_of_datasets_for_machine_learning_research&oldid=819203835.

[31] S. Cho, F. Zhang, and C. Edwards, "Anomaly detection for controlled lagrangian particles," in *OCEANS 2017 - Anchorage*, 2017, pp. 1–6.

[32] D. Park, Z. Erickson, T. Bhattacharjee, and C. C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 407–414, ISBN: 978-1-4673-8026-3. [Online]. Available: http://ieeexplore.ieee.org/document/7487160/.

[33] S. Ha and S. Choi, "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors," *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 381–388, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7727224/.

[34] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers.," *Sensors (Basel, Switzerland)*, vol. 10, no. 2, pp. 1154–75, 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22205862http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3244008.

[35] M. Balazia and P. Sojka, "Gait Recognition from Motion Capture Data," *ACM Trans. Multimedia Comput. Commun. Appl. ACM Transactions on Multimedia Computing Communications and Applications*, vol. 1, no. 18, 2017. [Online]. Available: https://doi.org/10.1145/nnnnnnn.nnnnnnn.

[36] K. ELLIS, J. KERR, S. GODBOLE, J. STAUDENMAYER, and G. LANCKRIET, "Hip and Wrist Accelerometer Algorithms for Free-Living Behavior Classification," *Medicine & Science in Sports & Exercise*, vol. 48, no. 5, pp. 933–940, 2016. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/26673126http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4833514http://content.wkhealth.com/linkback/openurl?sid=WKPTLP:landingpage&an=00005768-201605000-00022.

[37] *Avigilon - Avigilon Previews the Future of its Self-Learning Video Analytics*. [Online]. Available: http://ir.avigilon.com/Investor-Relations/Press-Releases/Press-Releases-Details/2017/Avigilon-Previews-the-Future-of-its-Self-Learning-Video-Analytics/.

[38] N. Kiryati, T. R. Raviv, Y. Ivanchenko, and S. Rochel, "Real-time Abnormal Motion Detection in Surveillance Video," [Online]. Available: http://people.csail.mit.edu/tammy/Publications/AbnMotionIcpr08.pdf.

[39] J. Feng-Shun Lin, V. Joukov, and D. Kuli, "Human Motion Segmentation by Data Point Classification," 2014. [Online]. Available: https://ece.uwaterloo.ca/~dkulic/pubs/LinEMBC2014.pdf.

[40] L. P, A. Lasa, K. Umezawa, Y. Nakamura, and A. Billard, "Learning Robot Skills Through Motion Segmentation and Constraints Extraction," [Online]. Available: http://lasa.epfl.ch/publications/uploadedFiles/ALP_WS_HRI2013.pdf.

[41] *Introduction to Anomaly Detection*. [Online]. Available: https://www.datascience.com/blog/python-anomaly-detection.

[42] C. Sample and G. Jones, "Introduction to Anomaly Detection," 2013. [Online]. Available: https://resources.sei.cmu.edu/asset_files/Presentation/2013_017_001_51271.pdf.

[43] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv. Article*, vol. 41, no. 10, 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882.

[44] *ISO 20958:2013 - Condition monitoring and diagnostics of machine systems – Electrical signature analysis of three-phase induction motors*, 2013. [Online]. Available: https://www.iso.org/standard/39839.html.

[45] Fluke Corporation, *Fluke Connect® Condition Monitoring Subscription - Vibration*, 2018. [Online]. Available: https://connect.fluke.com/en/stores/product/fluke-connect-condition-monitoring-vibration-software-subscription.

[46] *Valmet Corporate Website*. [Online]. Available: http://www.valmet.com/automation-solutions/condition-monitoring/.

[47] Brüel & Kjær, "Measuring Vibration," [Online]. Available: https://www.bksv.com/media/doc/br0094.pdf.

[48] Q. Wei, X. Zhang, Y. Wang, N. Feng, and Y. Shen, "Rail defect detection based on vibration acceleration signals," in *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, IEEE, 2013, pp. 1194–1199, ISBN: 978-1-4673-4623-8. [Online]. Available: http://ieeexplore.ieee.org/document/6555602/.

[49] I. Donelson J. and R. Dicus, "Bearing defect detection using on-board accelerometer measurements," *ASME/IEEE Joint Railroad Conference*, pp. 95–102, 2002.

[50] B. Scannell, "Enabling Continuous and Reliable Process Monitoring with Wireless Vibration Sensors," [Online]. Available: www.analog.com.

[51] E. Cochran, C. Miller, and P. Bullemer, "Abnormal Situation Management in petrochemical plants: can a Pilot's Associate crack crude?" In *Proceedings of the IEEE 1996 National Aerospace and Electronics Conference NAECON 1996*, vol. 2, IEEE, pp. 806–813, ISBN: 0-7803-3306-3. [Online]. Available: http://ieeexplore.ieee.org/document/517744/.

[52] O. Pettersson, "Execution monitoring in robotics: A survey," *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188900500134X?via%3Dihub.

[53] A. Jain and C. C. Kemp, "Improving robot manipulation with data-driven object-centric models of everyday forces," *Autonomous Robots*, vol. 35, no. 2, pp. 143–159, 2013. [Online]. Available: https://doi.org/10.1007/s10514-013-9344-1.

[54] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, 2017. [Online]. Available: https://doi.org/10.21105%2Fjoss.00205.

[55] Phidgets Inc., *Phidgets Sensors Products*, 2018. [Online]. Available: https://www.phidgets.com/?tier=0{\&}catid=3{\&}pcid=12.

[56] Seeed Studio. (2018). Grove System, [Online]. Available: http://wiki.seeedstudio.com/Grove_System/ (visited on 04/05/2018).

[57] L3 Aviation Products, *What is a Flight Data Recorder (FDR)? L3 Aviation Products*, 2018. [Online]. Available: http://www.l3aviationproducts.com/faq-items/fdr/.

[58] Honeywell Aerospace, *Health and Usage Monitoring - HUMS — Honeywell Aerospace*, 2018. [Online]. Available: https://aerospace.honeywell.com/en/product-listing/health-and-usage-monitoring.

[59] F. Ruffier and N. Franceschini, "Optic flow regulation: the key to aircraft automatic guidance," *Robotics and Autonomous Systems*, vol. 50, pp. 177–194, 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.816&rep=rep1&type=pdf.

[60] S. B. Fuller, M. Karpelson, A. Censi, K. Y. Ma, and R. J. Wood, "Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli," [Online]. Available: http://dx.doi.org/10.1098/rsif.2014.0281http://rsif.royalsocietypublishing.org..

[61] A Mohamed, S Watkins, R Clothier, M Abdulrahim, K Massey, and R Sabatini, "Fixed-wing MAV attitude stability in atmospheric turbulencePart 2: Investigating biologically-inspired sensors," *Progress in Aerospace Sciences*, vol. 71, pp. 1–13, 2014. [Online]. Available: http://ac.els-cdn.com/S0376042114000621/1-s2.0-S0376042114000621-main.pdf?_tid=7989eba6-806a-11e7-9e6d-00000aab0f27&acdnat=1502658226_4a0da51e050cc452f37c221b66ea2c50.

[62] N. Speal, *Bag2csv.py*, https://gist.github.com/marc-hanheide/4c35796e6a7cd0042dca274bf9e5e9f5, 2013.

[63] J. Pappalardo, *The Search for Aliens Starts Nowin Antarctica*. [Online]. Available: https://www.popularmechanics.com/space/solar-system/a14572143/europa-icefin-aliens-antarctica/.

[64] M. Fralick, *A Campus Without Boundaries: Under Ice*, 2018. [Online]. Available: https://www.gtalumni.org/s/1481/alumni/17/gt-magazine.aspx?sid=1481&gid=21&pgid=13398&cid=29982&ecid=29982&crid=0&calpgid=13302&calcid=29985.

[65] "iRobot ® Create ® 2 Open Interface (OI) Specification based on the iRobot ® Roomba ® 600," [Online]. Available: http://anrg.usc.edu/ee579/spring2016/Roomba/iRobot_Roomba_600_Open_Interface_Spec.pdf.

[66] I. Phidgets, *PhidgetSpatial Precision 3/3/3 High Resolution - 1044_0B at Phidgets*. [Online]. Available: https://www.phidgets.com/?tier=3&catid=10&pcid=8&prodid=1038.

[67] Open Source Robotics Foundation, *ROS.org — Powering the world's robots*. [Online]. Available: http://www.ros.org/.

[68] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," [Online]. Available: http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf.

[69] Resin.io, *Home — Resin.io*. [Online]. Available: https://resin.io/.

[70] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 –56.

[71] *Python Data Analysis Library pandas: Python Data Analysis Library*. [Online]. Available: https://pandas.pydata.org/.